KTH Computer Science
and Communication

Short Proofs May Be Spacious: Understanding Space in Resolution

JAKOB NORDSTRÖM

Doctoral Thesis
Stockholm, Sweden, 2008

TRITA-CSC-A 2008:05
ISSN-1653-5723
ISRN-KTH/CSC/A--08/05--SE
ISBN 978-91-7178-936-5

Skolan för datavetenskap och kommunikation
Kungliga Tekniska högskolan
SE-100 44 Stockholm
SVERIGE / SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i datalogi fredagen den 9 maj 2008 kl 10.00 i sal D3, Lindstedtsvägen 5, Kungliga Tekniska högskolan, Stockholm.

Tryck: Universitetsservice US-AB

Abstract

Most state-of-the-art satisfiability algorithms today are variants of the DPLL procedure augmented with clause learning. The two main bottlenecks for such algorithms are the amounts of time and memory used. Thus, understanding time and memory requirements for clause learning algorithms, and how these requirements are related to one another, is a question of considerable practical importance.

In the field of proof complexity, these resources correspond to the length and space of resolution proofs for formulas in conjunctive normal form (CNF). There has been a long line of research investigating these proof complexity measures and relating them to the width of proofs, another measure which has turned out to be intimately connected with both length and space. Formally, the length of a resolution proof is the number of lines, i.e., clauses, the width of a proof is the maximal size of any clause in it, and the space is the maximal number of clauses kept in memory simultaneously if the proof is only allowed to infer new clauses from clauses currently in memory.

While strong results have been established for length and width, our understanding of space has been quite poor. For instance, the space required to prove a formula is known to be at least as large as the needed width, but it has remained open whether space can be separated from width or whether the two measures coincide asymptotically. It has also been unknown whether the fact that a formula is provable in short length implies that it is also provable in small space (which is the case for length versus width), or whether on the contrary these measures are "completely unrelated" in the sense that short proofs can be maximally complex with respect to space.

In this thesis, as an easy first observation we present a simplified proof of the recent length-space trade-off result for resolution in (Hertel and Pitassi 2007) and show how our ideas can be used to prove a couple of other exponential trade-offs in resolution.

Next, we prove that there are families of CNF formulas that can be proven in linear length and constant width but require space growing logarithmically in the formula size, later improving this exponentially to the square root of the size. These results thus separate space and width. Using a related but different approach, we then resolve the question about the relation between space and length by proving an optimal separation between them. More precisely, we show that there are families of CNF formulas of size $\mathrm{O}(n)$ that have resolution proofs of length $\mathrm{O}(n)$ and width $\mathrm{O}(1)$ but for which any proof requires space $\Omega(n/\log n)$. All of these results are achieved by studying so-called pebbling formulas defined in terms of pebble games over directed acyclic graphs (DAGs) and proving lower bounds on the space requirements for such formulas in terms of the black-white pebbling price of the underlying DAGs.

Finally, we observe that our optimal separation of space and length is in fact a special case of a more general phenomenon. Namely, for any CNF formula F and any Boolean function $f : \{0,1\}^d \mapsto \{0,1\}$, replace every variable x in F by $f(x_1, \ldots, x_d)$ and rewrite this new formula in CNF in the natural way, denoting the resulting formula $F[f]$. Then if F and f have the right properties, $F[f]$ can be proven in resolution in essentially the same length and width as F but the minimal space needed for $F[f]$ is lower-bounded by the number of variables that have to be mentioned simultaneously in any proof for F.

Keywords: Proof complexity, resolution, space, length, width, separation, lower bound, pebble game, pebbling formula

Sammanfattning

Om man ser på de bästa nu kända algoritmerna för att avgöra satisfierbarhet hos logiska formler så är de allra flesta baserade på den så kallade DPLL-metoden utökad med klausulinlärning. De två viktigaste gränssättande faktorerna för sådana algoritmer är hur mycket tid och minne de använder, och att förstå sig på detta är därför en fråga som har stor praktisk betydelse.

Inom området beviskomplexitet svarar tids- och minnesåtgång mot längd och minne hos resolutionsbevis för formler i konjunktiv normalform (CNF-formler). En lång rad arbeten har studerat dessa mått och även jämfört dem med bredden av bevis, ett annat mått som visat sig höra nära samman med både längd och minne. Mer formellt är längden hos ett bevis antalet rader, dvs. klausuler, bredden är storleken av den största klausulen, och minnet är maximala antalet klausuler som man behöver komma ihåg samtidigt om man under bevisets gång bara får dra nya slutsatser från klausuler som finns sparade.

För längd och bredd har man lyckats visa en rad starka resultat men förståelsen av måttet minne har lämnat mycket i övrigt att önska. Till exempel så är det känt att minnet som behövs för att bevisa en formel är minst lika stort som den nödvändiga bredden, men det har varit en öppen fråga om minne och bredd kan separeras eller om de två måtten mäter “samma sak” i den meningen att de alltid är asymptotiskt lika stora för en formel. Det har också varit okänt om det faktum att det finns ett kort bevis för en formel medför att formeln också kan bevisas i litet minne (motsvarande påstående är sant för längd jämfört med bredd) eller om det tvärtom kan vara så att längd och minne är “helt orelaterade” på så sätt att även korta bevis kan kräva maximal mängd minne.

I denna avhandling presenterar vi först ett förenklat bevis av trade-off-resultatet för längd jämfört med minne i (Hertel och Pitassi 2007) och visar hur samma idéer kan användas för att visa ett par andra exponentiella avvägningar i relationerna mellan olika beviskomplexitetsmått för resolution.

Sedan visar vi att det finns formler som kan bevisas i linjär längd och konstant bredd men som kräver en mängd minne som växer logaritmiskt i formelstorleken, vilket vi senare förbättrar till kvadratroten av formelstorleken. Dessa resultat separerar således minne och bredd. Genom att använda andra men besläktade idéer besvarar vi därefter frågan om hur minne och längd förhåller sig till varandra genom att separera dem på starkast möjliga sätt. Mer precist visar vi att det finns CNF-formler av storlek $\mathrm{O}(n)$ som har resolutionbevis i längd $\mathrm{O}(n)$ och bredd $\mathrm{O}(1)$ men som kräver minne minst $\Omega(n/\log n)$. Det gemensamma temat för dessa resultat är att vi studerar formler som beskriver stenläggningsspel, eller pebblingspel, på riktade acykliska grafer. Vi bevisar undre gränser för det minne som behövs för den så kallade pebblingformeln över en graf uttryckt i det svart-vita pebblingpriset för grafen i fråga.

Slutligen observerar vi att vår optimala separation av minne och längd i själva verket är ett specialfall av en mer generell sats. Låt F vara en CNF-formel och $f : \{0,1\}^d \mapsto \{0,1\}$ en boolesk funktion. Ersätt varje variabel x i F med $f(x_1, \ldots, x_d)$ och skriv om denna nya formel på naturligt sätt som en CNF-formel $F[f]$. Då gäller, givet att F och f har rätt egenskaper, att $F[f]$ kan bevisas i resolution i väsentligen samma längd och bredd som F, men att den minimala mängd minne som behövs för $F[f]$ är åtminstone lika stor som det minimala antalet variabler som måste förekomma samtidigt i ett bevis för F.

Nyckelord: Beviskomplexitet, resolution, minne, längd, bredd, separation, undre gräns, pebblingspel, pebblingformel

Acknowledgements

First of all, I want to thank my advisor Johan Håstad. It has been a real privilege to have Johan as my advisor. I have learnt so much from him in so many ways. To the extent that I now know anything about this strange and wonderful art that is called research, it is very much thanks to him. I am grateful to Johan for patiently giving feed-back during this work, shooting down innumerable buggy ideas, encouraging me to pursue less flawed proof attempts, and also providing some invaluable insights of his own. It is an honour for me that this led to that I can now name Johan not only my advisor, but also my co-author.

Another person who has been very important for this thesis is my second co-author Eli Ben-Sasson. I am thankful to Eli for mentioning an open question in one of his papers that, as it turned out, became the guiding theme of my whole PhD project. I am glad that he encouraged me to keep working on the problem when I had managed to make some partial progress, and I am very happy that he invited me to the Technion where we finally solved it—just in time for thesis submission.

My former PhD student colleague Jonas Holmerin deserves to be acknowledged for a number of different reasons. I had a great time with Jonas already during our undergraduate studies at Stockholm University, and even more so later when I joined him as a PhD student at the Royal Institute of Technology (KTH). In this context, though, Jonas should be acknowledged most importantly for tipping me off about the excellence PhD position at KTH which funded my research and made this thesis possible. Also, I want to express my gratitude to the foundations *Johan och Jakob Söderbergs stiftelse* and *Sven och Dagmar Saléns stiftelse* for grants supporting the final stage of my PhD project.

From the Theory group at KTH, I wish to thank Douglas Wikström for interesting discussions about computer science, life and everything. I am grateful to Per Austrin, Mikael Goldmann, and Gunnar Kreitz for generous feedback during various stages of this work. Also, a big thanks to Joel Brynielsson for being my local tech guru, gently introducing me to (among other things) the mysterious METAPOST program that has been used to draw all the figures in this thesis. I have enjoyed the company of many other colleagues at the Royal Institute of Technology, and at the risk of forgetting some I would nevertheless like to especially mention Mads Dam, Isaac Elias, Lars Engebretsen, Gustav Hast, Fredrik Niemelä, Anna Palbom, and Mårten Trolin.

Outside of the Royal Institute of Technology, I would like to thank Paul Beame, Maria Klawe, Philipp Hertel, and Toniann Pitassi for valuable discussions concerning their work, and Nicholas Pippenger for answering questions about pebbling. I am grateful to Dieter van Melkebeek for very helpful comments during the process of writing up the journal version of my first paper that improved the presentation considerably and also taught me much about how to write papers. Also, I want to thank Nathan Segerlind for help with references regarding proof complexity in general and clause learning in particular, and Jan Krajíček and Jacobo Torán for clarifying some last-minute questions as this thesis was about to go to press.

During my visit to Israel towards the very end of my PhD studies, I greatly enjoyed talking to Arie Matsliah and Amir Yehudayoff about different research problems. Also, I wish to thank Lance Fortnow and Madhu Sudan for finding the time to talk with me during their visits.

Due to the very fortunate outcome of the research visit to Israel, this thesis has been prepared in great haste. I am most obliged to Per Austrin, Jonas Holmerin, Johan Håstad, Malin Sandström, Nathan Segerlind, Mårten Trolin, and Douglas Wikström for helping me with proof-reading parts of the thesis, catching bugs, suggesting various improvements, and providing insightful comments in general. Also, I am grateful to my friends Tom Airaksinen and Erik Arvidsson and my wife Elisabet for help with checking and improving the thesis on a somewhat less technical, but no less important, level. It goes without saying that the errors that remain are solely my own responsibility. A special thanks goes to the KTH CSC janitor Julio Cabrera for invaluable practical assistance during the printing process.

Finally, concluding the professional part of the acknowledgements, I want to mention that Albert Atserias suggested to me already in 2006, albeit in a slightly different context, that studying XOR-pebbling contradictions might be a possible way forward. I wish I would have worked a bit harder on that idea back then before giving up...

On a more personal note, I am most grateful to Sara Arvidsson and Veronica Westergård for helping out with some emergency baby-sitting when I was trying to meet a submission deadline during the spring of 2007 while being on part-time parental leave. In the end, I did not make it that time, but the insights I gained during those days of intense research laid the foundations for later results.

Mum and dad—how can I thank you enough? Thank you ever so much for all support and all practical help, especially during these last two years when Elisabet and I have been finishing our theses. I do not know what we would have done without you.

Daniel—you are truly the sunshine of our lives. Thanks for all the joy and happiness you bring us.

Elisabet—thank you for your patience, for your support, and for your love. I know it has been a hard time. Thank you for seeing this through. I would not have made it without you, and this thesis is dedicated to you. I love you.

God—I believe that you exist and that you somehow helped me to do this. Thank you.

Contents

List of Figures

Part I

Introduction

Chapter 1

A Popular Science Introduction

On June 4, 1996, the Ariane 5 rocket exploded less than a minute into its maiden voyage. The reason was a bug in the onboard navigation and guidance software, in a piece of program code designed for the predecessor Ariane 4 but useless for Ariane 5. To quote the inquiry board report, this code had been left in “presumably based on the view that, unless proven necessary, it was not wise to make changes in software which worked well on Ariane 4.” In other words, it was an instance of the old and trusted principle of software design: “if it ain’t broke, don’t fix it.”

It had not been taken into consideration, however, that the flight characteristics of Ariane 5 in the first 30 seconds of flight differed substantially from that of Ariane 4. This caused an overflow error, but since there had been a large safety margin for these values in Ariane 4, no error checking had been included to take care of a possible overflow. Instead, the program crashed and the ensuing error message was interpreted by the onboard computer as flight data. As a result, the rocket made “an abrupt course correction that was not needed, compensating for a wrong turn that had not taken place” [43], broke up and exploded. As an extra absurdity, the piece of software containing the bug actually served no purpose even on Ariane 4 once the rocket was in the air, but had been designed to keep running during the first 40 seconds or so of the flight as a “special feature”.

At a cost of more than 7 billion US dollars, the first Ariane 5 launch was arguably one of the most expensive firework displays in human history. But although spectacular, the Ariane 5 failure is just one of many examples of a widespread problem in the software industry. Large software systems tend to become highly complex, with intricate interdependencies in the code which makes it hard or impossible to analyze them and predict their behaviour. For less dramatic examples of this we do not have to go further than ourselves. Most of us have probably experienced, at one time or another, how the computer just “freezes” and refuses to respond, for instance. This is a symptom of exactly the same problem.

The contemporary hardware industry experiences analogous difficulties. Perhaps the most well-known illustration of this is the embarrassing flaw in Intel’s

Pentium microprocessor discovered in October 1994: in very rare cases, the processor got the answer wrong when dividing two numbers (see, for instance, [58] for a readable account). After first having tried to downplay the problem, Intel was later forced to offer replacements to customers at a cost of 475 million US dollars.

Intel has not been alone in having problems. The growing complexity of state-of-the-art hardware devices is outpacing the capacity of the tools used to check that they are correct. Making things worse, there is an increasing pressure to keep the development time of new devices to a minimum to reduce the time-to-market. As a consequence, components under development are more likely to contain errors, while less time can be spent on *validation*, that is, making sure that what has been built corresponds to the intended design. And no matter how much time is set aside for simulation and testing, this can never provide full coverage of all possible cases in increasingly complex designs.

So what can be done? This chapter briefly outlines one approach, and in doing so describes how I came in contact with this kind of problems while doing my Master's thesis [59] at Prover Technology (`www.prover.com`) and how I subsequently moved on to do a PhD thesis in proof complexity at the Royal Institute of Technology.

Writing popular science is a tricky task. It involves simplifying and generalizing, not seldom to the point of stating things that formally are just not true. Thus, the reader is warned that what is written here is almost, but not quite, entirely unlike my actual research [1]. The thesis proper starts in Chapter 2. This introductory chapter is an attempt to tell friends and family, whom I do not expect to read Chapter 2 onwards, at least something about what I have been doing these years. The text is intended to be understandable for those with no prior knowledge of computer science. (I know it does get slightly technical towards the end, though, but I hope that at least some of you will make it all the way.)

1.1 Let's Get Formal

How can one master the complexity of large technical systems? One proposed solution to this problem is the adoption of *formal methods* in software and hardware design. The Free On-line Dictionary of Computing (`www.foldoc.org`) describes formal methods as:

> *Mathematically based techniques for the specification, development and verification of software and hardware systems.*

The idea is that if the methods for specifying and designing systems are formalized, it becomes possible to use mathematical tools to *prove* that the designed system conforms to its specification. Formal methods have formerly been the subject of mainly academic study, but applied research in and usage of formal methods have increased greatly during the last two decades.

Here comes a very much simplified example: Suppose that we would like to verify that a processor divides two numbers correctly. All the processor does is to work with ones and zeros, corresponding to true and false, and we can therefore describe its design in mathematical logic. Also, we can certainly describe in mathematical logic that for any numbers x and y, we want the processor to output x/y and nothing else. Now we can write all of this as one big mathematical formula with a lot of variables saying, essentially, "the processor looks like this, and division of two numbers means this, and the processor just described accurately divides two numbers." If this formula is true, we have designed our processor correctly. If it is not true, then we want to find a concrete counter-example for which values of the variables the formula turns out to be false. Hopefully, this could give us a clue to where in the design we made a mistake.

The formulas that one gets in this way are typically huge, so trying to work them out with pen and paper is generally not a good idea. What we want is a computer program that takes a formula as input and either confirms that the formula is true or produces a counter-example. Such a program is called an *automated theorem prover*. Formal verification is one area where such programs are useful, but the list of applications is much larger than hardware and software design analysis and verification. Automated theorem provers are also used among other things for scheduling problems, in artificial intelligence research, and even to prove results in theoretical mathematics.

Needless to say, the above description provides only a very simplistic view, but we hope that it serves the purpose of conveying the general idea. A nice (and more accurate, though still informal) introduction to formal methods in Swedish is [66].

1.2 Proving Things Is Complex

So far this seems to be very good news. We start with a hard problem, reformulate it using some mathematical modelling, and then leave the tedious detail-checking to the computer. So does this mean that we have now solved the problem?

Well, not quite. The bad news is that proving logical formulas seems to be a very difficult task for a computer. On the face of it, it might appear obvious what one should do: just let the computer check all possible cases. Is this not exactly the kind of monotone routine work at which computers excel? The problem is that there are just too many cases to check. Suppose that the formula has N variables. Each variable can be either true or false, so all in all we get 2^N different cases. And if our formula contains, say, one million variables (which could typically be the case for real-world problems), this means that we get $2^{1000000}$ cases. This is a number with more than 300,000 digits. To understand how large this number is, consider that even if every atom in the universe was a modern supercomputer that had been running at full speed ever since the beginning of time some 13.7 billion years ago, all of them together would only have covered a completely negligible fraction of these cases by now. So we really would not have time to wait for them to finish...

Intriguingly, although there are many advanced automated theorem provers that perform really well most of the time, for all of them there appear to exist very hard formulas for which they are not able to do anything essentially better than checking all cases. What is more, it seems that for any computer program, or *algorithm*, that takes a formula as input and checks if it is correct, there will exist formulas that are just too hard (as we noted above, checking all cases is not a feasible alternative). We write *seems*, because no one knows if this is really so or if we just have not found out yet how to construct smart enough algorithms. In fact, this is one of the big, deep, open problems in modern mathematics. It has been named one of the seven challenges for the new millennium by the Clay Mathematics Institute [57] and it carries a one million dollar reward for whoever solves it.

So proving logical formulas seems to be a hard problem. In proof complexity, we want to understand better exactly *how hard* this problem is (not necessarily aiming for the one million dollars, though). The way we do this is to, perhaps somewhat unexpectedly, try to get rid of the formula-proving algorithms altogether.

Every automated theorem prover uses some kind of rules when it proves a formula, and it is no restriction to ask the algorithm to write down which rules it uses, and how, in order to reach its conclusions. The result is that when an algorithm has proven a formula, there will also be a written *proof* explaining why the formula is true. Different algorithms work in different ways, and the way the algorithm works determines what rules can be used in its proofs. In this way, every automated theorem prover can be made to correspond to a so-called *proof system.* In proof complexity, we are interested in studying proof systems.

What can such proof systems tell us? If a formula that we want to prove is very small, there might exist small proofs for it. If the formula is large, it seems natural to expect that the proofs will have to be large as well. What we are interested in is measuring how fast the proof size grows *as a function of the size of the formula.* If there are reasonably small proofs for all formulas measured in terms of formula size, then maybe we can hope to find these proofs quickly. But if we know that the minimal proof size grows very rapidly in a proof system, then there is no hope of being able to prove formulas in an efficient way. How can we know this? Well, even if the algorithm is lucky and finds the best proof, we know that all proofs—even this one—must contain a very large number of steps. Hence, the algorithm will have to run for a very long time before it is done going through all of the steps in this proof.

In other words, the sizes of proofs in a proof system can tell us something interesting about the performance of the corresponding automated theorem prover. Studying proof systems is also interesting for other, more theoretical reasons, but that is outside the scope of this introduction.

1.3 What Is a Proof?

The discussion of proofs and proof systems in the preceding section somehow turned out to be very abstract. What is a "proof system"? And what do we mean by a "proof"? In this section, we try to be more concrete.

Indeed, what is a proof? This question is almost impossible to answer. For instance, this thesis is full of (claimed) proofs. Some of them are presented in painstaking detail, with page after page of technical proof steps. In other proofs, we just assert that certain statements are “obvious” or “straightforward to verify” and leave the details to the reader. If one wants to start an animated discussion among theoretical computer scientists, one good way might be to ask what exactly is the right level of detail when presenting a proof. There seem to be as many views on this subject as there are theoretical computer scientists.

Fortunately, we do not have to give an all-encompassing answer to this question. We just have to provide a definition that works for the very restricted formal systems that we study in proof complexity. And here it turns out that the right definition is essentially the following: *“A proof is something with the help of which you can easily verify a statement that you might not otherwise know how to check.”*

This thesis deals with proof systems for logic. However, the notation in such systems can appear intimidating and takes some time to get used to. We want to give an example that can be understood right away. Therefore we will instead concentrate on numbers, namely the integers $1, 2, 3, 4, \ldots$

We start with some facts that are not related to proof complexity, but which might be interesting anyway. Any integer greater than 1 can be “decomposed” into a product of integers (also greater than 1). For instance, we can write $77 = 7 \cdot 11$ and $30 = 2 \cdot 3 \cdot 5$. One of the basic results in mathematics says that if we decompose a number into “minimal components”, in the sense that the integers in the product cannot themselves be written as products of other integers, then this decomposition, or *factorization*, is unique. This is known as the *Fundamental Theorem of Arithmetic.*

In the examples given above, it is easy to check that $2, 3, 5, 7$, and 11 cannot be written as products of smaller integers, so the decompositions are minimal. Now of course, we can always write $77 = 7 \cdot 11 = 11 \cdot 7$ in two different ways, but apart from this kind of reordering the factorization is unique. The “minimal components” in a factorization are called *prime numbers*, and they just “decompose” into a product of one number, i.e., themselves.

Consider now the following simple problem: we are given an integer N and we are asked to tell what its factorization is. For concreteness, let us think of the two numbers $N_1 = 25957$ and $N_2 = 510510$. How can they be written as products of smaller integers? Or maybe they are prime numbers? (The reader might wish to have a go at solving these cases before reading on—one of them is easy but the other is slightly harder.)

Simple as it may look, this problem is at the forefront of current research in computer science. Enormous amounts of time and money have been invested in trying to come up with good algorithms for factorization. So far, no one has succeeded. Or at least, so we believe. It is a well-known secret that governmental agencies such as the National Defence Radio Establishment (FRA) in Sweden and the National Security Agency in the US are working hard on this problem as well, but they do not publish their results.

Why all this interest? Because many (probably, most) researchers believe that this problem is so hard that there cannot exist any efficient factorization algorithms. And although we do not know for sure that this is so, we can construct IT security systems for which the security rests on the assumption that factorization is hard. Much of modern cryptography is built in this way. And this is not only theoretical science. Chances are that, for instance, the security of your Internet bank is based on the belief that factoring large integers, or solving some similar problem, is so hard that it is practically impossible to do even on a very powerful computer.

Now back to proof complexity. Let us look at the following variant of the problem discussed above: We are given integers N and k from a person claiming that the prime number factorization of N contains exactly k numbers. For instance, suppose someone tells us that $N_1 = 25957$ is the product of 2 prime numbers. Can we decide whether this is true? What is a proof that would convince us?

It seems we will have a hard time deciding this on our own since we do not know how to factor integers efficiently. But since the person we are talking to apparently knows the factors, we can ask him or her to give a proof simply by listing them. If as an answer we get the list

$$\begin{array}{r} 101 \\ 257 \end{array}$$

then we can check that this proof is correct fairly easily. First, the list has two elements, just as claimed. Second, if we multiply them the result is indeed 25957. Third, we can also verify that neither 101 nor 257 can be written as a product of smaller integers, that is, they are prime numbers. And if the number N is larger, so that we cannot work this out by hand, there are efficient algorithms both for multiplying numbers and for checking prime numbers and we can let a computer verify the proof for us.

We want to highlight two important features of a proof system. First, *we can never be fooled into accepting a false statement.* Suppose the person had instead told us that 25957 has 6 factors. We ask for the proof, and get, perhaps, the list

$$\begin{array}{r} 2 \\ 2 \\ 3 \\ 3 \\ 7 \\ 103 \end{array}$$

This is of course close—the product is 25956, just off by one—but we have no problems to discover this and reject the purported proof as false.

Second, *there is always a proof for true statements.* Of course, it happens that proofs are buggy (though, let us hope, not in this thesis). For example, if someone says that the two factors of 25957 are

$$\begin{array}{r} 100 \\ 258 \end{array}$$

we cannot accept this as a proof since it is plainly wrong. But the point is that since it is in fact true that 25957 has exactly 2 factors, there is another, correct proof that can convince us that this is so (which in this case is the first proof given above). *Finding* such a proof might be wholly another matter, but at least it *exists.*

Once we have a proof system, we want to understand its properties. Two important properties, that are the focus of this thesis, are the *length* and *space* of proofs. Consider our second number $N_2 = 510510$, which we claim has 7 factors. The proof of this is the list

$$\begin{array}{r} 2 \\ 3 \\ 5 \\ 7 \\ 11 \\ 13 \\ 17 \end{array}$$

which has 7 lines, so we will say that the length of the proof is 7.

The space of a proof is, roughly, the number of things we need to remember while verifying the proof. To check the proof above, we can multiply the numbers one by one and remember how many numbers we have multiplied so far. Then all we need to keep track of is (1) the number of factors so far, (2) the accumulated product so far, and (3) the factor we are looking at right now. When we are done going through the list, we check that we have seen 7 factors and that their product is 510510. If so, everything is in order and we accept the proof. Since we need to remember 3 things only, the space of the proof is 3.

Note that even though the proof we get may be very long (if the integer has a large number of factors), all proofs in our proof system will be "simple" in the sense that we only need space 3 to verify them, no matter how long the proofs are. For more general proof systems, this will typically not be the case, and it is an interesting question how length and space are related to one another.

This thesis studies the proof system *resolution* for logical formulas, which is important since it is the basis for the best automated theorem provers currently known. An open question concerning resolution has been the following:

> *Suppose that a proof is short. Is it then true that the proof is also easy in the sense that it can be verified in small space? Or can it be that although the proof is short, it has such intricate structure that one has to remember essentially the whole proof while verifying it?*

The main result in the thesis says that there are formulas that have short proofs, but where these proofs must by necessity be so complicated that one cannot just check them "locally" line by line (as we did in our proof system above). Instead, one has to keep almost the whole proof in memory while verifying it. That is, we prove what is called a *lower bound*, in this case a lower bound saying that certain formulas must always require a lot of space.

Apart from being interesting in the context of proof complexity, the lower bound that we prove could possibly have implications for automated theorem provers as well. It might be interpreted as meaning that there are formulas which are very easy to prove if you do it "the right way", but finding this easy proof is a little bit like looking for a needle in a haystack. However, there are subtleties involved here that are beyond the scope of this introduction. We refer to Section 12.3 for a more detailed discussion of these issues.

1.4 Why Is It Hard to Prove Lower Bounds?

Proving lower bounds is a very special endeavour. In most of computer science, and in science in general, one shows how to actually *do* things: construct, compute, prove, invent... Establishing a lower bound means showing that something *cannot* be done.

For instance, if we want to show that a certain problem can be solved efficiently on a computer, it is clear what we need to do: come up with an algorithmic idea, write a computer program, and prove that the solution is correct. In contrast, to prove that a problem *cannot* be efficiently solved using computers, we need to show that no program, no matter how it is constructed, can solve the problem faster than some specified time bound. Note that we cannot make any assumptions on what the computer programs might be up to. For all we know, they can be completely wild. Maybe there is an ingenious solution to our problem out there that no one has thought of before. Maybe there is a program that does crazy things but somehow solves the problem very quickly. We cannot *assume* that this is not the case—we have to *prove* it.

In this final section, we present a case study aimed at illustrating some of these difficulties. Again, we avoid logic since we want an example that is reasonably accessible without prior knowledge of computer science or higher mathematics. Instead, we will discuss how to sort numbers efficiently. The problem we want the computer to solve is the following:

> *Given a list of N distinct integers in any order, write a list of these integers in ascending order.*

We understand that sorting 1000 numbers will probably have to take longer time than sorting just 10 numbers. What we are interested in is proving lower bounds on the time needed expressed as a function of the number of integers N in the problem input. As a running example when describing different approaches, we will use the following list

$$670,\ 45,\ 490,\ 642,\ 475,\ 124,\ 802,\ 266$$

of eight numbers to be sorted.

One natural approach, that one can use for instance when one is dealt cards from a deck of playing cards and want to sort them, is to keep a set of unsorted cards to the right and then insert the cards one by one into a sorted set of cards on

1.	**670**		45	490	642	475	124	802	266
2.	**45**	670		490	642	475	124	802	266
3.	45	**490**	670		642	475	124	802	266
4.	45	490	**642**	670		475	124	802	266
5.	45	**475**	490	642	670		124	802	266
6.	45	**124**	475	490	642	670		802	266
7.	45	124	475	490	642	670	**802**		266
8.	45	124	**266**	475	490	642	670	802	

Figure 1.1: Example of insertion sort (sorted numbers left, unsorted numbers right).

the left. This algorithm is known as *insertion sort*, and Figure 1.1 illustrates how it works for our example set of numbers.

How much time does insertion sort take? It is a fast algorithm for small set of integers, but when the size of the list grows, performance starts to deteriorate. When we are sorting the last integers in the list, we might have to scan through a large part of the integers already sorted before finding the right place where the current one should be inserted. Look, for instance, at stage 8 in Figure 1.1 when we are sorting the final number 266. Note that the computer has no way of immediately "seeing" where this number should go. From the computer's point of view, 266 is just some pattern of zeros and ones that can be compared with other patterns of zeros and ones representing other numbers. So we have to scan either from the left or from the right, and if the needed position is somewhere in the middle this will take us almost $N/2$ steps. In can be shown that this can be expected to happen for a reasonably large fraction of the numbers, and therefore insertion sort takes time quadratic in the number of integers to be sorted, i.e., time proportional to N^2.

But maybe this is the best we can hope for anyway? There are N numbers to sort, and we need to know how all numbers are related to one another in order to sort them in the correct order. Since we have N numbers, this means that there are $N(N-1) \approx N^2$ distinct *pairs* of numbers, so we would expect that we have to do something like N^2 comparisons before we know how all these pairs are related. So, counting only the comparisons and ignoring other things that the algorithm might also need to do, we see that a lower bound on the time to sort numbers must be proportional to N^2, right?

Wrong, as it turns out. Let us sketch a completely different algorithmic idea. Suppose that we instead split our list of number into two halves $670, 45, 490, 642$ and $475, 124, 802, 266$. Suppose furthermore, using some wishful thinking, that we can sort these halves quickly to get $45, 490, 642, 670$ and $124, 266, 475, 802$. Then we can very efficiently merge these two sorted lists into one larger sorted list. Namely, we start by looking at the first element in both lists. 45 is smallest, so it goes first. Now 124 is smaller than 490, so it goes second, and in the same way we pick 266 and 475 from the second list. Then, since 802 is larger than 490, we switch to the first list again and remove all numbers from it, since they are all smaller than 802. Finally, we add 802 to the end of our larger list, which is now sorted. Note that as opposed to for insertion sort, we could immediately find the right place in the list for all numbers without having to scan through the list of previously sorted problems.

Of course, there seems to be a catch here. How do we sort the smaller lists to begin with? Well, why not use the same idea? Split the first list $670, 45, 490, 642$ into two sublists $670, 45$ and $490, 642$, sort them, and merge. But that is begging the question—how do we sort these sublists? Well, these sublists only have two elements, so they are very easy to sort: just swap the numbers if they are in the wrong order, and otherwise do nothing. And if we keep splitting larger lists into smaller lists, sooner or later we will get very small lists that we can immediately sort in this way.

This might seem like a very strange idea, but it works. And not only does it work, but it is dramatically much faster than insertion sort. This algorithm is called *merge sort* for understandable reasons, and although it is far from obvious (and we will not try to prove it here) it runs in time proportional to $N \ln N$ instead of N^2. For large data sets, this difference means seconds instead of hours or days.

Fine, so now we are down from N^2 to $N \ln N$ for the time needed to do sorting. Can we sort even faster? Or is $N \ln N$ the correct answer? Let us try to prove a lower bound.

As we noted above, we do not know how the algorithm might work, so let us focus on something that we know it will have to do, namely, compare the numbers. Let us just count the comparisons and give a lower bound for the number of such operations. We can represent all possible outcomes of the comparisons made by drawing a graph. See Figure 1.2 for an example of an algorithm sorting three elements. Inside the ellipses we write which comparisons are made, and the branches show the two possible outcomes. We get what is called a *tree* in computer science jargon (although computer scientists insist on drawing their trees upside down).

Any sorting algorithm corresponds to such a tree in the following way. The algorithm starts by making some comparison, which we indicate at the *root* of the tree (which is, consequently, at the top of the figure). Depending on the result of the comparison, the algorithm can choose to do one of two different things, which corresponds to moving along either of the two branches. As soon as the algorithm knows the answer, it stops and outputs the sorted list. We mark this in the tree by adding a *leaf* with the corresponding sorting order. Since this tree can be seen as a help to *decide* which sorting order is the right one, it is called a *decision tree.*

We will use this decision tree to prove our lower bound. We do this by making three observations.

- First, the number of comparisons that the algorithm will make in the worst case is the longest path to a leaf along branches in the tree. The list of integers can be presented to us in any order, so all paths in the decision tree are possible outcomes for the algorithm.

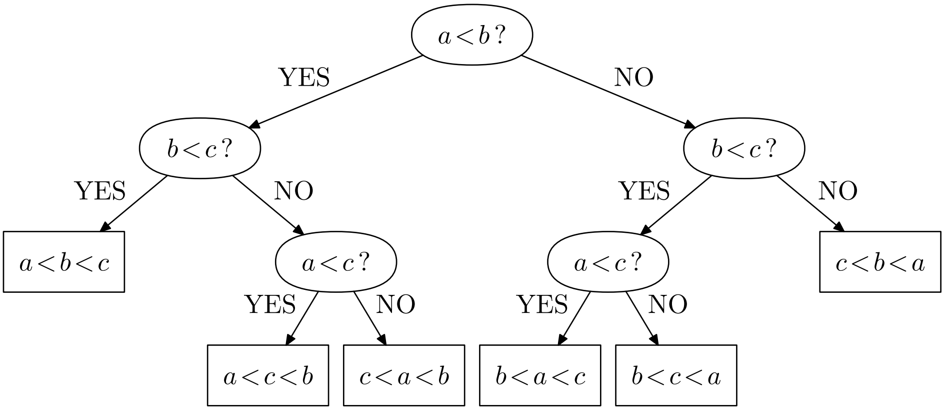


Figure 1.2: Decision tree for sorting three numbers a, b, c.

- Second, there must exist at least one leaf in the decision tree for every possible answer. Note again that the list of integers can be presented to us in any order, so all orderings of the numbers are possible candidate answers. Since all the answers the algorithm gives are found in the leaves, there must be a leaf for every possible case.
- Third, the number of possible different answers when sorting three integers is $3 \cdot 2 \cdot 1 = 6$. Any of the 3 integers in the list can be the smallest one. Once we know the position of the smallest integer, there are 2 possible positions left for the middle integer, and then there is only 1 choice left for the position of the largest integer. In general, when we are sorting N integers, there are

  $$N(N-1)(N-2)\ldots \text{ (continue multiplying with smaller integers) } \ldots 3\cdot2\cdot1$$

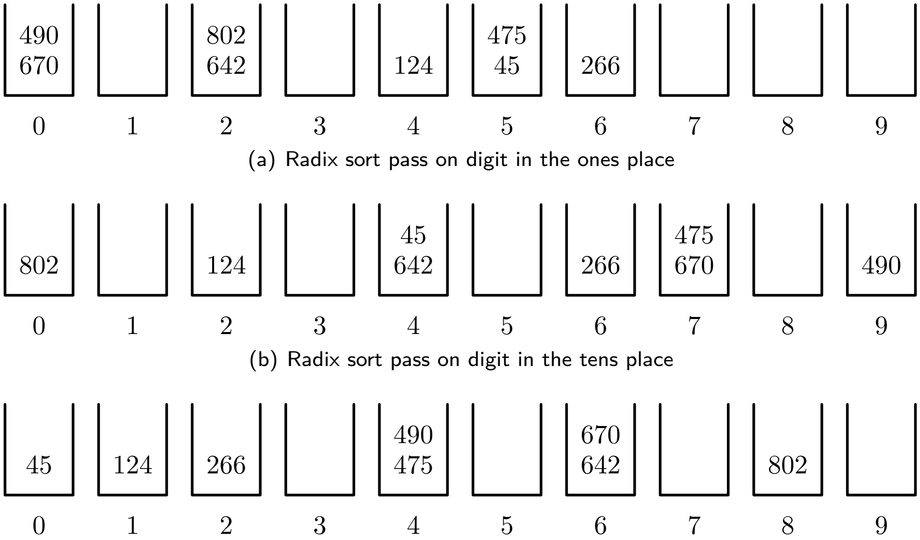
  possible different answers, since the smallest integer in the input list can be in any one of the N positions, the next smallest integer can be in any of the $N-1$ remaining positions, after that the third smallest integer can be in any of the $N-2$ remaining positions, et cetera.

By combining these observations, we can establish a lower bound in two steps.

In the first step, it can be shown (and this is not hard) that a decision tree with L leaves must have some path that is at least roughly $\ln L$ long. If all paths would be shorter than that, there simply would not be room in the tree for L distinct leaves. As a concrete example, note that Figure 1.2 proves that it is impossible to sort 3 numbers by just 2 comparisons. Why? Because there are 6 possible answers, but if all paths in the tree would have length at most 2, then we could only have $4 < 6$ leaves.

In the second step, it can be proven (and this is harder) that

$$\ln\big(N(N-1)(N-2)\cdots 3\cdot 2\cdot 1\big) \approx N \ln N \ .$$



(a) Radix sort pass on digit in the ones place

(b) Radix sort pass on digit in the tens place

(c) Radix sort pass on digit in the hundreds place

Figure 1.3: Example of radix sort of list 670, 45, 490, 642, 475, 124, 802, 266.

But this means that since there are $N(N-1)(N-2)\cdots 3\cdot 2\cdot 1$ distinct possible answers for the correct sorting order that all need their own distinct leaves, any sorting algorithm will have to do $N \ln N$ comparisons in the worst case. This must clearly be a lower bound on time. (Remember that we are ignoring a lot of other things that the algorithm will also have to do.) Thus, the merge sort algorithm that we presented above is an optimal sorting algorithm!

Almost. Indeed, $N \ln N$ is a lower bound for so-called comparison-based sorting algorithms. But what about a sorting algorithm that does not do comparisons? A sorting algorithm that does not compare the things that it is sorting might sound slightly odd, but remember that we cautioned in the beginning of this section that the algorithms that we are fighting against when proving lower bounds can be crazy.

So here is a crazy algorithm. It works as follows (this description will probably be easier to understand by looking at Figure 1.3). We create ten *buckets* labelled 0, 1, 2, 3, …, 8, 9. Since our numbers have three digits, we make three passes over all the numbers. In the first pass, we sort all numbers by the digit in the ones place. That is, 670 is placed in bucket 0, 45 is placed in bucket 5, 490 is placed in bucket 0 on top of 670, 642 is placed in bucket 2, 475 is placed in bucket 5 on top of 45, et cetera (see Figure 1.3(a)). Then we turn all buckets upside down, pour out the numbers, and go through the numbers again from left to right, and from top to bottom in each column. This time, however, we sort them by the digit in the tens place, so 670 is placed in bucket 7, 490 is placed in bucket 9, 642 is placed in bucket 4, 802 is placed in bucket 0, et cetera (see Figure 1.3(b) for the result of this second pass). In the third pass, we turn all buckets upside down and go through the numbers again from left to right, sorting them by the digit in the hundreds place. Since 45 has no hundreds digit, we place it in bucket 0. This leads to the numbers ending up in buckets as in Figure 1.3(c). Finally, we turn the buckets upside down and list all numbers from left to right, and in the same bucket from top to bottom (after turning the bucket upside down, remember). It is easy to check that listing the numbers in Figure 1.3(c) in this way, we get our eight numbers sorted in correct order.

This algorithm is called *radix sort*. It might seem slightly magical, but it actually works. What is more, it beats merge sort (and our lower bound) by running in time proportional to N, not $N \ln N$.

What about N? Is *that* at least a lower bound on the time? Yes it is. For in order to be able to sort N numbers, any algorithm must, in some way, *look* at each number at least once. Since there are N numbers in the list we get, this must take time proportional to N. (And this time, the statement can be made into a formal proof.)

Concluding this introductory chapter, I hope that I have been able to convey to the reader in Sections 1.1 and 1.2 at least something of what proof complexity is and what it might be good for. I hope that the end of Section 1.3 gave some vague flavour of the kind of problems that I have been studying, and that the more technical stuff in Section 1.4 provided some indication of why proving lower bounds is not entirely trivial. (And if you found this interesting, there is the whole rest of the thesis just waiting to be read. . .)

Chapter 2

Formal Introduction and Results Overview

Ever since the fundamental NP-completeness result of Cook [31], the problem of deciding whether a given propositional logic formula in conjunctive normal form (CNF) is satisfiable or not has been on center stage in Theoretical Computer Science. In more recent years, SATISFIABILITY has gone from a problem of mainly theoretical interest to a practical approach for solving applied problems. Although all known Boolean satisfiability solvers (SAT-solvers) have exponential running time in the worst case, enormous progress in performance has led to satisfiability algorithms becoming a standard tool for solving a large number of real-world problems such as hardware and software verification, experiment design, circuit diagnosis, and scheduling.

Perhaps a somewhat surprising aspect of this development is that the most successful SAT-solvers to date are still variants of the Davis-Putnam-Logemann-Loveland (DPLL) procedure [36, 37] augmented with *clause learning*, which produces proofs in the resolution proof system. For instance, the great majority of the best algorithms at the 2007 round of the international SAT competitions [75] fit this description.

DPLL procedures perform a recursive backtrack search in the space of partial truth value assignments. The idea behind clause learning, or *conflict-driven learning*, is that at each failure (backtrack) point in the search tree, the system derives a reason for the inconsistency in the form of a new clause and then adds this clause to the original CNF formula ("learning" the clause). This can save much work later on in the proof search, when some other partial truth value assignment fails for similar reasons.

The main bottleneck for this approach, other than the obvious one that the worst-case running time can be exponential, is the amount of memory used by the algorithms. Since there is only a finite amount of memory, all clauses cannot be stored. The difficulty lies in obtaining a highly selective and efficient clause caching scheme that nevertheless keeps the clauses needed. Thus, understanding time and memory requirements for clause learning algorithms, and how these requirements are related to each other, is a question of great practical importance. We refer to, for instance, [14, 48, 73] for a more detailed discussion of clause learning (and SAT-solving in general) with examples of applications.

The study of proof complexity originated with the seminal paper of Cook and Reckhow [33]. In its most general form, a proof system for a language L is a predicate $P(x, \pi)$, computable in time polynomial in $|x|$ and $|\pi|$, having the property that for all $x \in L$ there is a string π (a *proof*) for which $P(x, \pi) = 1$, whereas for any $x \notin L$ it holds for all strings π that $P(x, \pi) = 0$. A proof system is said to be polynomially bounded if for every $x \in L$ there exists a proof π_x for x that has size at most polynomial in $|x|$. A *propositional proof system* is a proof system for the language of tautologies in propositional logic.

From a theoretical point of view, one important motivation for proof complexity is the intimate connection with the fundamental question of P versus NP. Since NP is exactly the set of languages with polynomially bounded proof systems, and since TAUTOLOGY can be seen to be the dual problem of SATISFIABILITY, we have the famous theorem of [33] that $\mathsf{NP} = \mathsf{co\text{-}NP}$ if and only if there exists a polynomially bounded propositional proof system. Thus, if it could be shown that there are no polynomially bounded proof systems for propositional tautologies, $\mathsf{P} \neq \mathsf{NP}$ would follow as a corollary since P is closed under complement. One way of approaching this distant goal is to study stronger and stronger proof systems and try to prove superpolynomial lower bounds on proof size. However, although great progress has been made in the last couple of decades for a variety of propositional proof systems, it seems that we are still very far from fully understanding the reasoning power of even quite simple ones.

Another important motivation is that, as was mentioned above, designing efficient algorithms for proving tautologies (or, equivalently, testing satisfiability), is a very important problem not only in the theory of computation but also in applied research and industry. All automated theorem provers, regardless of whether they actually produce a written proof, explicitly or implicitly define a system in which proofs are searched for and rules which determine what proofs in this system look like. Proof complexity analyzes what it takes to simply write down and verify the proofs that such an automated theorem-prover might find, ignoring the computational effort needed to actually find them. Thus a lower bound for a proof system tells us that any algorithm, even an optimal (non-deterministic) one making all the right choices, must necessarily use at least the amount of a certain resource specified by this bound. In the other direction, theoretical upper bounds on some proof complexity measure give us hope of finding good proof search algorithms with respect to this measure, provided that we can design algorithms that search for proofs in the system in an efficient manner. For DPLL procedures with clause learning, the time and memory resources used are measured by the *length* and *space* of proofs in the resolution proof system.

The field of proof complexity also has rich connections to cryptography, artificial intelligence and mathematical logic. Some good books and survey papers providing more details are [12, 15, 28, 30, 76, 83].

2.1 Previous Work

Any formula in propositional logic can be converted to a CNF formula that is only linearly larger and is unsatisfiable if and only if the original formula is a tautology. Therefore, any sound and complete system for refuting CNF formulas can be considered as a general propositional proof system.

Perhaps the single most studied proof system in propositional proof complexity, *resolution*, is such a system that produces proofs of the unsatisfiability of CNF formulas. The resolution proof system appeared in [22] and began to be investigated in connection with automated theorem proving in the 1960s [36, 37, 72]. Because of its simplicity—there is only one derivation rule—and because all lines in a proof are clauses, this proof system readily lends itself to proof search algorithms.

Being so simple and fundamental, resolution was also a natural target to attack when developing methods for proving lower bounds in proof complexity. In this context, it is most straightforward to prove bounds on the *length* of refutations, i.e., the number of clauses, rather than on the total size of refutations. The length and size measures are easily seen to be polynomially related. In 1968, Tseitin [81] presented a superpolynomial lower bound on refutation length for a restricted form of resolution, called *regular* resolution, but it was not until almost 20 years later that Haken [44] proved the first superpolynomial lower bound for general resolution. This weakly exponential bound of Haken has later been followed by many other strong results, among others truly exponential lower bounds on resolution refutation length for different formula families in, for instance, [13, 21, 29, 82].

A second complexity measure for resolution, first made explicit by Galil [41], is the *width*, measured as the maximal size of a clause in the refutation. Ben-Sasson and Wigderson [21] showed that the minimal width $W(F \vdash 0)$ of any resolution refutation of a k-CNF formula F is bounded from above by the minimal refutation length $L(F \vdash 0)$ by

$$W(F \vdash 0) = \mathrm{O}\bigl(\sqrt{n \log L(F \vdash 0)}\bigr) \ , \tag{2.1}$$

where n is the number of variables in F. Since it is also easy to see that resolution refutations of polynomial-size formulas in small width must necessarily be short (simply for the reason that $(2 \cdot \#\text{variables})^w$ is an upper bound on the total number of distinct clauses of width w), the result in [21] can be interpreted as saying roughly that there exists a short refutation of the k-CNF formula F if and only if there exists a (reasonably) narrow refutation of F. This interpretation also gives rise to a natural proof search heuristic: to find a short refutation, search for refutations in small width. It was shown in [19] that there are formula families for which this heuristic exponentially outperforms any DPLL procedure regardless of branching function.

The formal study of the *space* measure in resolution was initiated by Esteban and Torán [39, 79]. Intuitively, the space $Sp(\pi)$ of a resolution refutation π is the maximal number of clauses one needs to keep in memory while verifying the refutation, and the space $Sp(F \vdash 0)$ of refuting F is defined as the minimal space of any resolution refutation of F. A number of upper and lower bounds for refutation space in resolution and other proof systems were subsequently presented in, for example, [4, 18, 38, 40]. Just as for width, the minimum space of refuting a formula can be upper-bounded by the size of the formula. Somewhat unexpectedly, however, it also turned out that the lower bounds on resolution refutation space for several different formula families exactly matched previously known lower bounds on refutation width. Atserias and Dalmau [10] showed that this was not a coincidence, but that the inequality

$$W(F \vdash 0) \leq \mathit{Sp}(F \vdash 0) + \mathrm{O}(1) \tag{2.2}$$

holds for any k-CNF formula F, where the (small) constant term depends on k.

The space measure discussed above is known as *clause space*. A less well-studied space measure, introduced by Alekhnovich et al. [4], is *variable space*, which counts the maximal number of variable occurrences that must be kept in memory simultaneously. Ben-Sasson [16] used this measure to obtain a trade-off result for clause space versus width in resolution, proving that there are k-CNF formulas F_n that can be refuted in constant clause space and constant width, but for which any refutation π_n must have $\mathit{Sp}(\pi_n) \cdot W(\pi_n) = \Omega(n/\log n)$. More recently, Hertel and Pitassi [45] showed that there are CNF formulas F_n for which any refutation of F_n in minimal variable space $\mathit{VarSp}(F_n \vdash 0)$ must have exponential length, but by adding just 3 extra units of storage one can instead get a resolution refutation in linear length.

2.2 Two Questions Left Open by Previous Research

Despite all the research that has gone into understanding the resolution proof system, a number of fundamental questions have remained unsolved. We discuss two such questions, which have been the main focus of our research, below.

Firstly, what is the relation between clause space and width? The inequality (2.2) says that (essentially) clause space $\geq$ width, but it leaves open whether this relationship is tight or not. Do refutation clause space and width always coincide, or is there a formula family that separates the two measures asymptotically?

Secondly, what is the relation between clause space and length? For width, rewriting the bound in (2.1) in terms of the number of clauses $|F_n|$ instead of the number of variables we get that that if the width of refuting F_n is $\omega\bigl(\sqrt{|F_n| \log |F_n|}\bigr)$, then the length of refuting F_n must be superpolynomial in $|F_n|$. This is known to be almost tight, since [25] shows that there is a k-CNF formula family $\{F_n\}_{n=1}^{\infty}$ with $W(F_n \vdash 0) = \Omega\bigl(\sqrt[3]{|F_n|}\bigr)$ but $L(F_n \vdash 0) = \mathrm{O}(|F_n|)$. Hence, formula families refutable in polynomial length can have somewhat wide minimum-width refutations, but not arbitrarily wide ones.

For clause space, the inequality (2.2) tells us that any correlation between length and clause space cannot be tighter than the correlation between length and width, so in particular we get from the previous paragraph that k-CNF formulas refutable in polynomial length may have at least "somewhat spacious" minimum-space refutations. At the other end of the spectrum, given any resolution refutation π of F in length L it can be proven using results from [39, 46] that $Sp(\pi) = \mathrm{O}(L/\log L)$. This gives an upper bound on any possible separation of the two measures. But is there a Ben-Sasson–Wigderson kind of upper bound on clause space in terms of length similar to (2.1)? For all that we know, it might hold that $Sp(F_n \vdash 0) = \mathrm{O}\bigl(\sqrt{|F_n| \log L(F_n \vdash 0)}\bigr)$. Or are length and space on the contrary unrelated in the sense that there exist k-CNF formulas F_n with short refutations but maximal possible refutation space $Sp(F_n \vdash 0) = \Omega\bigl(L(F_n \vdash 0)/\log L(F_n \vdash 0)\bigr)$ in terms of length? We note that for the restricted case of so-called tree-like resolution, [39] showed that there is a tight correspondence between length and clause space, exactly as for length versus width.

These two questions have been discussed in, for instance, [16, 38, 40, 76, 80], but there seems to have been no consensus on what the right answer should be. However, these papers identify a plausible formula family for answering the question, namely so-called *pebbling contradictions* defined in terms of pebble games over directed acyclic graphs. Non-constant lower bounds on the space of refuting pebbling contradictions would separate space and width and possibly also clarify the relation between space and length if the bounds were good enough. On the other hand, a constant upper bound on the refutation space would improve the trade-off results for different proof complexity measures for resolution in [16].

2.3 Our Contribution

In this thesis, we answer both questions above. Our first result is a separation of space and width, which we obtain by proving an asymptotically tight bound on space for pebbling contradictions over binary trees. More precisely, the results can be stated as follows (formal definitions are given in Chapters 4 and 5).

Theorem 2.1 ([60]). *The clause space of refuting pebbling contradictions over complete binary trees of height h in resolution grows as $\Theta(h)$, provided that the number of variables per vertex in the pebbling contradictions is at least* 2.

Corollary 2.2 ([60]). *For all $k \geq 4$, there is a family $\{F_n\}_{n=1}^{\infty}$ of k-CNF formulas of size $\mathrm{O}(n)$ that can be refuted in width $W(F_n \vdash 0) = \mathrm{O}(1)$ but require space $Sp(F_n \vdash 0) = \Theta(\log n)$.*

We then further strengthen this result by establishing an asymptotically tight bound on the clause space of refuting pebbling contradictions over so-called pyramid graphs.

Theorem 2.3 ([63]). *The clause space of refuting pebbling contradictions over pyramid graphs of height h in resolution grows as $\Theta(h)$, provided that the number of variables per vertex in the pebbling contradictions is at least* 2.

This yields the first separation of space and length (in the sense of a polynomial lower bound on space for formulas refutable in polynomial length) that is not a consequence of a corresponding lower bound on width, as well as an exponential improvement of the separation of space and width in Corollary 2.2.

Corollary 2.4 ([63]). *For all $k \geq 4$, there is a family $\{F_n\}_{n=1}^{\infty}$ of k-CNF formulas of size $\Theta(n)$ that can be refuted in resolution in length $L(F_n \vdash 0) = \mathrm{O}(n)$ and width $W(F_n \vdash 0) = \mathrm{O}(1)$ but require clause space $Sp(F_n \vdash 0) = \Theta(\sqrt{n})$.*

Finally, by studying a slightly modified version of pebbling contradictions defined in terms of exclusive or over the variables, we achieve lower bounds on clause space in terms of pebbling price for any DAG G.

Theorem 2.5 ([20]). *The space of refuting XOR-pebbling contradictions over any DAG G in resolution is lower-bounded by the black-white pebbling price of G, provided that the number of variables per vertex in the XOR-pebbling contradictions is at least 2.*

As an immediate corollary, we get the strongest possible separation of clause space and length in resolution.

Corollary 2.6 ([20]). *For all $k \geq 6$ there is a family $\{F_n\}_{n=1}^{\infty}$ of k-CNF formulas of size $\Theta(n)$ that can be refuted in resolution in length $L(F_n \vdash 0) = \mathrm{O}(n)$ and width $W(F_n \vdash 0) = \mathrm{O}(1)$ but require clause space $Sp(F_n \vdash 0) = \Omega(n/\log n)$.*

This separation is asymptotically optimal since, as was noted above, a refutation in length $\mathrm{O}(n)$ is always possible to carry out in space $\mathrm{O}(n/\log n)$. For width, it is not hard to show that a formula of size $\mathrm{O}(n)$ is refutable in width $\mathrm{O}(n)$, so the separation of space and width is also very nearly optimal, except for perhaps a logarithmic factor.

We mention that in ongoing research briefly reviewed in Chapter 11, we show that our optimal separation of space and length is in fact a special case of a more general phenomenon. Informally, for any CNF formula F and any non-constant Boolean function $f : \{0,1\}^d \mapsto \{0,1\}$, we can substitute every variable x in F by the function $f(x_1, \ldots, x_d)$ evaluated on d new variables $x_1, \ldots, x_d$. This yields a new Boolean formula , which we can rewrite in conjunctive normal form in the natural way. Let us denote the resulting CNF formula by $F[f]$. Then we can prove the following.

Theorem 2.7. *If $f : \{0,1\}^d \mapsto \{0,1\}$ is any non-constant Boolean function for d fixed, and if F is any CNF formula with $L(F \vdash 0) = \mathrm{O}(n)$ and $W(F \vdash 0) = \mathrm{O}(1)$, then it holds that $L\bigl(F[f] \vdash 0\bigr) = \mathrm{O}(n)$ and $W\bigl(F[f] \vdash 0\bigr) = \mathrm{O}(1)$. Furthermore, if f has the property that no single variable x_i can fix $f(x_1, \ldots, x_d)$ to true or false, then the clause space $Sp\bigl(F[f] \vdash 0\bigr)$ of refuting $F[f]$ is lower-bounded by the number of variables that have to be mentioned simultaneously in any proof for F.*

With hindsight, Theorem 2.5 and Corollary 2.6 can be shown to follow as more or less immediate consequences of Theorem 2.7.

In addition to the sequence of lower bound results stated above, which are the main contribution of this thesis, we also make the observation that the proof of the trade-off result in [45] can be greatly simplified, and the parameters slightly improved. Using similar ideas, we can also prove exponential trade-offs for length with respect to clause space and width. Namely, we show that there are k-CNF formulas such that if we insist on finding the resolution refutation in smallest clause space or smallest width, respectively, then we have to pay with an exponential increase in length. We state the theorem only for length versus clause space.

Theorem 2.8 ([61]). *There is a family of k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that:*

- *The clause space of refuting F_n in resolution is $Sp(F_n \vdash 0) = \Theta\big(\sqrt[3]{n}\big)$.*
- *Any resolution refutation $\pi : F_n \vdash 0$ in minimal clause space must have length $L(\pi) = \exp\big(\Omega\big(\sqrt[3]{n}\big)\big)$.*
- *There are resolution refutations $\pi' : F_n \vdash 0$ in asymptotically minimal clause space $Sp(\pi') = \mathrm{O}\big(Sp(F_n \vdash 0)\big)$ and length $L(\pi') = \mathrm{O}(n)$, i.e., linear in the formula size.*

Using pattern matching on the proof of Theorem 2.8, it is possible to show a trade-off result for length versus width on exactly the same form.

Chapter 3

High-Level Overview of Tools and Methods

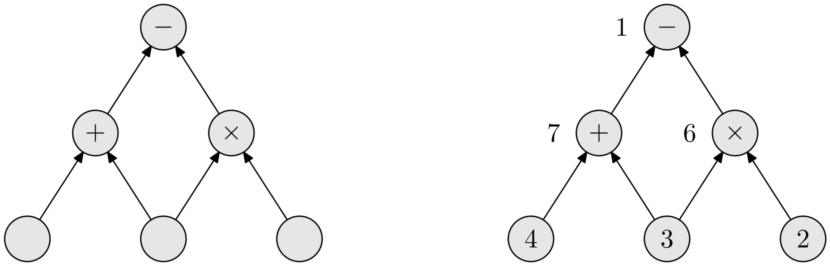
We now try to give an intuitive, high-level description of the proofs of our results. Although the technical details in the different proofs vary, there are some recurring themes in the constructions that we want to make explicit. We conclude the chapter by outlining how the rest of this thesis is organized.

3.1 Sketch of Preliminaries

A *resolution refutation* of a CNF formula F can be viewed as a sequence of derivation steps on a blackboard. In each step we may write a clause from F on the blackboard (an *axiom* clause), erase a clause from the blackboard or derive some new clause implied by the clauses currently written on the blackboard.[1] The refutation ends when we reach the contradictory empty clause. The *length* of a resolution refutation is the number of distinct clauses in the refutation, the *width* is the size of the largest clause in the refutation, and the *clause space* is the maximum number of clauses on the blackboard simultaneously. We write $L(F \vdash 0)$, $W(F \vdash 0)$ and $Sp(F \vdash 0)$ to denote the minimum length, width and clause space, respectively, of any resolution refutation of F.

The *pebble game* played on a DAG G models the calculation described by G, where the source vertices contain the input and non-source vertices specify operations on the values of the predecessors (see Figure 3.1). Placing a pebble on a vertex v corresponds to storing in memory the partial result of the calculation described by the subgraph rooted at v. Removing a pebble from v corresponds to deleting the partial result of v from memory. A *pebbling* of a DAG G is a sequence of moves starting with the empty graph G and ending with all vertices in G empty except for a pebble on the (unique) sink vertex. The *cost* of a pebbling is the maximal number of pebbles used simultaneously at any point in time during the

[1] For our proof, it turns out that the exact definition of the derivation rule is not essential—our lower bound holds for any sound rule. What is important is that we are only allowed to derive new clauses that are implied by the set of clauses currently on the blackboard.



(a) DAG encoding calculation. (b) After pebbling with results filled in.

Figure 3.1: Example of modelling calculation as pebbling of DAG.

pebbling. The *pebbling price* of a DAG G is the minimum cost of any pebbling, i.e., the minimum number of memory registers required to perform the complete calculation described by G.

The pebble game on a DAG G can be encoded as an unsatisfiable CNF formula Peb_G^d, a so-called *pebbling contradiction* of degree d. See Figure 3.2 for a small example. Very briefly, pebbling contradictions are constructed as follows:

- Associate d variables $x(v)_1, \ldots, x(v)_d$ with each vertex v (in Figure 3.2 we have $d = 2$).
- Specify that all sources have at least one true variable, for example, the clause $x(r)_1 \vee x(r)_2$ for the vertex r in Figure 3.2.
- Add clauses saying that truth propagates from predecessors to successors. For instance, for the vertex u with predecessors r and s, clauses 4–7 in Figure 3.2 are the CNF encoding of the implication $(x(r)_1 \vee x(r)_2) \wedge (x(s)_1 \vee x(s)_2) \rightarrow (x(u)_1 \vee x(u)_2)$.
- To get a contradiction, conclude the formula with $\overline{x(z)}_1 \wedge \cdots \wedge \overline{x(z)}_d$ where z is the sink of the DAG.

We will need the fact, proven in [19], that a pebbling contradiction of degree d over a graph with n vertices of at most constant indegree can be refuted by resolution in length $\mathrm{O}\big(d^2 \cdot n\big)$ and width $\mathrm{O}(d)$.

3.2 Proof Idea for Pebbling Contradictions

Pebble games have been used extensively as a tool to prove time and space lower bounds and trade-offs for computation. Loosely put, a lower bound for the pebbling price of a graph says that although the computation that the graph describes can be performed quickly, it requires large space. Our hope is that when we encode pebble games in terms of CNF formulas, these formulas should inherit the same properties

$$
\begin{aligned}
& (x(r)_1 \lor x(r)_2) \\
\land\ & (x(s)_1 \lor x(s)_2) \\
\land\ & (x(t)_1 \lor x(t)_2) \\
\land\ & (\overline{x(r)}_1 \lor \overline{x(s)}_1 \lor x(u)_1 \lor x(u)_2) \\
\land\ & (\overline{x(r)}_1 \lor \overline{x(s)}_2 \lor x(u)_1 \lor x(u)_2) \\
\land\ & (\overline{x(r)}_2 \lor \overline{x(s)}_1 \lor x(u)_1 \lor x(u)_2) \\
\land\ & (\overline{x(r)}_2 \lor \overline{x(s)}_2 \lor x(u)_1 \lor x(u)_2) \\
\land\ & (\overline{x(s)}_1 \lor \overline{x(t)}_1 \lor x(v)_1 \lor x(v)_2) \\
\land\ & (\overline{x(s)}_1 \lor \overline{x(t)}_2 \lor x(v)_1 \lor x(v)_2) \\
\land\ & (\overline{x(s)}_2 \lor \overline{x(t)}_1 \lor x(v)_1 \lor x(v)_2) \\
\land\ & (\overline{x(s)}_2 \lor \overline{x(t)}_2 \lor x(v)_1 \lor x(v)_2)
\end{aligned}
$$

$$
\begin{aligned}
\land\ & (\overline{x(u)}_1 \lor \overline{x(v)}_1 \lor x(z)_1 \lor x(z)_2) \\
\land\ & (\overline{x(u)}_1 \lor \overline{x(v)}_2 \lor x(z)_1 \lor x(z)_2) \\
\land\ & (\overline{x(u)}_2 \lor \overline{x(v)}_1 \lor x(z)_1 \lor x(z)_2) \\
\land\ & (\overline{x(u)}_2 \lor \overline{x(v)}_2 \lor x(z)_1 \lor x(z)_2) \\
\land\ & \overline{x(z)}_1 \\
\land\ & \overline{x(z)}_2
\end{aligned}
$$

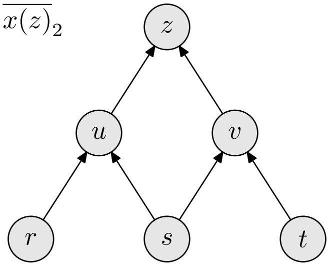


Figure 3.2: Pebbling contradiction $Peb^2_{\Pi_2}$ for the pyramid of height 2.

as the underlying graphs. That is, if we pick a DAG G with high pebbling price, since the corresponding pebbling contradiction encodes a calculation which needs a lot of memory we would like to try to argue that any resolution refutation of this formula should require large space. Then a separation result would follow since we already know that the formula can be refuted in short length.

More specifically, what we would like to do is to establish a connection between resolution refutations of pebbling contradictions on the one hand, and the so-called *black-white pebble game* [34] modelling the non-deterministic computations described by the underlying graphs on the other. Our intuition is that the resolution proof system should have to conform to the combinatorics of the pebble game in the sense that from any resolution refutation of a pebbling contradiction Peb^d_G we should be able to extract a pebbling of the DAG G.

Ideally, we would like to give a proof of a lower bound on the resolution refutation space of pebbling contradictions along the following lines:

1. First, find a natural interpretation of sets of clauses currently “on the blackboard” in a refutation of the formula Peb^d_G in terms of black and white pebbles on the vertices of the DAG G.

2. Then, prove that this interpretation of clauses in terms of pebbles captures the pebble game in the following sense: for any resolution refutation of Peb^d_G, looking at consecutive sets of clauses on the blackboard and considering the corresponding sets of pebbles in the graph we get a black-white pebbling of G in accordance with the rules of the pebble game.

3. Finally, show that the interpretation captures clause space in the sense that if the content of the blackboard induces N pebbles on the graph, then there must be at least N clauses on the blackboard.

Combining the above with known lower bounds on the pebbling price of G, this would imply a lower bound on the refutation space of pebbling contradictions and a separation from length and width. For clarity, let us spell out what the formal argument of this would look like.

Consider an arbitrary resolution refutation of Peb_G^d. From this refutation we extract a pebbling of G. At some point in time t in the obtained pebbling, there must be many pebbles on the vertices of G since this graph was chosen with high pebbling price. But this means that at time t, there are many clauses on the blackboard. Since this holds for any resolution refutation, the refutation space of Peb_G^d must be large. The separation result now follows from the fact that pebbling contradictions are known to be refutable in linear length and constant width if d is fixed.

Unfortunately, although this proof outline is intuitively appealing, we never quite get it to work. In the next section, we try to sketch how we attempt to implement the proof idea above, why it does not work, and what can be done to circumvent the problems.

3.3 Overview of Formal Proofs of Space Bounds

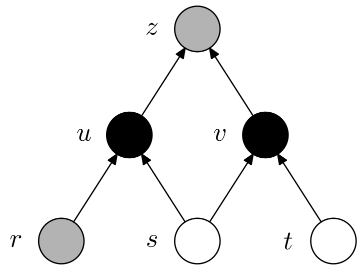
The black-white pebble game played on a DAG G can be viewed as a way of proving the end result of the calculation described by G. Black pebbles denote proven partial results of the computation. White pebbles denote assumptions about partial results which have been used to derive other partial results (i.e., black pebbles), but these assumptions will have to be verified for the calculation to be complete. The final goal is a black pebble on the sink z and no other pebbles in the graph, corresponding to an unconditional proof of the end result of the calculation with any assumptions made along the way having been eliminated. The *black-white pebbling price* $\mathsf{BW\text{-}Peb}(G)$ is the minimum cost of any black-white pebbling of G, or equivalently the minimum number of memory registers required for a non-deterministic computation of the calculation described by G.

Translating this to pebbling contradictions, it turns out that a fruitful way to think of a black pebble on v is that it should correspond to truth of the disjunction $\bigvee_{i=1}^{d} x(v)_i$ of all positive literals over v, or to "truth of v". A white pebble on a vertex w can be understood to mean that we need to *assume* the partial result on w to derive the black pebbles above w in the graph. Needing to assume the truth of w is the opposite of knowing the truth of w, so extending the reasoning above we get that a white-pebbled vertex should correspond to "falsity of w", i.e., to all negative literals $\overline{x(w)_i}$, $i \in [d]$, over w.

Using this intuitive correspondence, we can translate sets of clauses in a resolution refutation of Peb_G^d into black and white pebbles in G as in Figure 3.3. It

$$\begin{bmatrix} x(u)_1 \lor x(u)_2 \\ \overline{x(s)}_1 \lor \overline{x(t)}_1 \lor x(v)_1 \lor x(v)_2 \\ \overline{x(s)}_1 \lor \overline{x(t)}_2 \lor x(v)_1 \lor x(v)_2 \\ \overline{x(s)}_2 \lor \overline{x(t)}_1 \lor x(v)_1 \lor x(v)_2 \\ \overline{x(s)}_2 \lor \overline{x(t)}_2 \lor x(v)_1 \lor x(v)_2 \end{bmatrix}$$

(a) Clauses on blackboard.



(b) Corresponding pebbles in the graph.

Figure 3.3: Example of intuitive correspondence between clauses and pebbles.

is easy to see that if we assume $x(s)_1 \lor x(s)_2$ and $x(t)_1 \lor x(t)_2$, this assumption together with the clauses on the blackboard in Figure 3.3(a) imply $x(v)_1 \lor x(v)_2$, so v should be black-pebbled and s and t white-pebbled in Figure 3.3(b). The vertex u is also black since $x(u)_1 \lor x(u)_2$ certainly is implied by the blackboard. This translation from clauses to pebbles is arguably quite straightforward, and seems to yield well-behaved black-white pebblings for all "sensible" resolution refutations of Peb^d_G.

The problem is that we have no guarantee that the resolution refutations will be "sensible". Even though it might seem more or less clear how an optimal refutation of a pebbling contradiction should proceed, a particular refutation might contain unintuitive and seemingly non-optimal derivation steps that do not make much sense from a pebble game perspective. In particular, a resolution derivation has no obvious reason always to derive truth that is restricted to single vertices. For instance, it could add the axioms $\overline{x(u)}_i \lor \overline{x(v)}_2 \lor x(z)_1 \lor x(z)_2$, $i = 1, 2$, to the blackboard in Figure 3.3(a), derive that the truth of s and t implies the truth of either v or z, i.e., the clauses $\overline{x(s)}_i \lor \overline{x(t)}_j \lor x(v)_1 \lor x(z)_1 \lor x(z)_2$ for $i, j = 1, 2$, and then erase $x(u)_1 \lor x(u)_2$ from the blackboard, resulting in the set of clauses

$$\begin{bmatrix} \overline{x(s)}_1 \lor \overline{x(t)}_1 \lor x(v)_1 \lor x(z)_1 \lor x(z)_2 \\ \overline{x(s)}_1 \lor \overline{x(t)}_2 \lor x(v)_1 \lor x(z)_1 \lor x(z)_2 \\ \overline{x(s)}_2 \lor \overline{x(t)}_1 \lor x(v)_1 \lor x(z)_1 \lor x(z)_2 \\ \overline{x(s)}_2 \lor \overline{x(t)}_2 \lor x(v)_1 \lor x(z)_1 \lor x(z)_2 \end{bmatrix} . \tag{3.1}$$

Although it is hard to motivate from such a small example, this turns out to be a serious problem in that there appears to be no way that we can interpret such derivation steps in terms of black and white pebbles without making some component in the proof idea in Section 3.2 break down.

So what can we do? Well, if you can't beat 'em, join 'em! In order to prove Theorems 2.1, 2.3, and 2.5, we give up our attempts to translate resolution refutations into black-white pebblings and instead invent new pebble games (in three different flavours). These pebble games are on the one hand somewhat similar to the black-white pebble game, but on the other hand they have pebbling rules specifically designed so that tricky clause sets such as (3.1) can be interpreted in a satisfying way. Once we have done this, we construct bound proofs as outlined in Section 3.2 but using our modified pebble games instead of standard black-white pebbling.

The first step is to establish that for appropriate classes of graphs (of varying generality depending on which modified pebble game we use), any resolution refutation of a pebbling contradiction can be interpreted as a pebbling, but now in our modified game, of the DAG in terms of which this pebbling contradiction is defined. Intuitively, the reason that this works is that we have introduced auxiliary pebbling rules in the game that allow us to analyze apparently non-optimal derivation steps in the refutation. For instance, to prove Theorem 2.1 we use a game where one can not only place and remove pebbles, but also slide black pebbles downwards and white pebbles upwards. When we have set up the formal definitions in the right way, we can prove a "theorem template" of the following form.

Theorem Template 3.1. *Let Peb^d_G denote the pebbling contradiction of any degree $d \geq 1$ over a DAG G from some appropriate class of graphs. Then there is a translation function from sets of clauses derived from Peb^d_G into sets of black and white pebbles in G that translates any resolution refutation π of Peb^d_G into a pebbling $\mathcal{P}_\pi$ of G in our modified pebble game.*

The next step is to show that the number of pebbles used in $\mathcal{P}_\pi$ in Theorem Template 3.1 is related to the space of the resolution refutation π. It can be shown that we need at least $d \geq 2$ variables per vertex in order for such a bound to hold.

Theorem Template 3.2. *If π is a resolution refutation of a pebbling contradiction Peb^d_G of degree $d > 1$, then the cost of the associated pebbling $\mathcal{P}_\pi$ is bounded by the space of π by $\mathsf{cost}(\mathcal{P}_\pi) = \mathrm{O}(Sp(\pi))$.*

In fact, what we do is to apply some "reverse engineering" by defining pebbling cost in our modified pebble game precisely so that the bound in Theorem Template 3.2 holds.

Finally, we need lower bounds on pebbling price. Since the rules in our modified pebble game have been changed in comparison with those of the standard black-white pebble game, known bound on black-white pebbling price in the literature no longer apply. However, by using varying amounts of effort depending on the flavour of the modified pebble game, we can show for appropriate classes of graphs that the pebbling price in our modified pebble game is lower-bounded by black-white pebbling price.

Theorem Template 3.3. *If G is a DAG from some appropriate class of graphs, then the pebbling price of G in our modified pebble game is $\Omega\bigl(\mathsf{BW\text{-}Peb}(G)\bigr)$.*

Putting all of these components together, we can prove our main theorems and corollaries.

Theorem Template 3.4 (Space lower bound). *Let Peb_G^d denote the pebbling contradiction of degree $d > 1$ defined over a DAG G from some appropriate class of graphs. Then the clause space of refuting Peb_G^d by resolution is $Sp(Peb_G^d \vdash 0) = \Omega\bigl(\mathsf{BW\text{-}Peb}(G)\bigr)$.*

Proof sketch. Let π be any resolution refutation of Peb_G^d. Consider the associated pebbling $\mathcal{P}_\pi$ provided by Theorem Template 3.1. On the one hand, we know that $\mathsf{cost}(\mathcal{P}_\pi) = \mathrm{O}(Sp(\pi))$ by Theorem Template 3.2, provided that $d > 1$. On the other hand, Theorem Template 3.3 tells us that the cost of any pebbling of G is $\Omega\bigl(\mathsf{BW\text{-}Peb}(G)\bigr)$, so in particular we must have $\mathsf{cost}(\mathcal{P}_\pi) = \Omega\bigl(\mathsf{BW\text{-}Peb}(G)\bigr)$. Combining these two bounds on $\mathsf{cost}(\mathcal{P}_\pi)$, we see that $Sp(\pi) = \Omega\bigl(\mathsf{BW\text{-}Peb}(G)\bigr)$ and since this holds for any resolution refutation π the lower bound follows. □

Corollary Template 3.5 (Separation). *There is a family of k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that $L(F_n \vdash 0) = \mathrm{O}(n)$ and $W(F_n \vdash 0) = \mathrm{O}(1)$ but $Sp(F_n \vdash 0) = \Omega(g(n))$ for some suitable function $g(n) = \omega(1)$.*

Proof sketch. If we fix the pebbling degree $d \geq 2$ to some constant and pick $\{G_n\}_{n=1}^{\infty}$ to be a family of DAGs with $\Theta(n)$ vertices of bounded indegree, then the pebbling contradictions $Peb_{G_n}^d$ are a family of k-CNF formulas of size $\Theta(n)$ (compare with Figure 3.2). Also, when d is fixed the upper bounds mentioned at the end of Section 3.1 become $L\bigl(Peb_G^d \vdash 0\bigr) = \mathrm{O}(n)$ and $W\bigl(Peb_G^d \vdash 0\bigr) = \mathrm{O}(1)$. Corollary Template 3.5 now follows if we set $F_n = Peb_{G_n}^d$ for some suitable constant d and some family of DAGs $\{G_n\}_{n=1}^{\infty}$ for which our modified pebble game works, and then apply Theorem Template 3.4. □

3.4 Overview of Trade-off Results

Let us also quickly sketch the ideas (or tricks, really) used to prove our trade-off theorems for resolution.

We show the following version of the length-variable space trade-off theorem of Hertel and Pitassi [45], with somewhat improved parameters and a very much simpler proof.

Theorem 3.6. *There is a family of CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that:*

- *The minimal variable space of refuting F_n is $VarSp(F_n \vdash 0) = \Theta(n)$.*
- *Any refutation $\pi : F_n \vdash 0$ in minimal variable space has length $\exp(\Omega(\sqrt{n}))$.*
- *Adding at most 2 extra units of storage, it is possible to obtain a resolution refutation π' in variable space $VarSp(\pi') = VarSp(F_n \vdash 0) + 2 = \Theta(n)$ and length $L(\pi') = \mathrm{O}(n)$, i.e., linear in the formula size.*

The idea behind our proof is as follows. Take formulas G_n that are really hard for resolution and formulas H_m which have short refutations but require linear variable space, and set $F_n = G_n \wedge H_m$ for m chosen so that $\mathit{VarSp}\bigl(H_m \vdash 0\bigr)$ is only just larger than $\mathit{VarSp}\bigl(G_n \vdash 0\bigr)$. Then refutations in minimal variable space will have to take care of G_n, which requires exponential length, but adding one or two literals to the memory we can attack H_m instead in linear length.

The trade-off result in Theorem 2.8 for length versus clause space and its twin theorem for length versus width are shown using similar ideas.

3.5 Outline of This Thesis

The rest of this thesis is organized as follows. Chapter 4 gives a brief and selective introduction to proof complexity in general and resolution in particular, and reviews some known results. Chapter 5 describes pebble games and pebbling contradictions, and in Chapter 6 we present some bounds on pebbling price that we will need.

In Chapter 7, we give proofs of the trade-offs results in Theorems 2.8 and 3.6. In Chapter 8, we prove the space-width separation in Theorem 2.1 and Corollary 2.2, which is then improved to Theorem 2.3 and Corollary 2.4 in Chapter 9. Chapter 10 presents the optimal separation of space and length obtained by proving Theorem 2.5 and Corollary 2.6. Finally, in Chapter 11 we discuss how our results can be put in a more general framework, where the optimal separation is a simple corollary of Theorem 2.7.

We conclude the thesis in Chapter 12 by discussing a few questions that seem to be natural directions for future research in this area.

Part II

Background

Chapter 4

Proof Complexity and Resolution

In this chapter, we give a very brief overview of some of the central concepts in proof complexity. We then proceed to define the resolution proof system and state the results mentioned in Chapter 2, as well as some other results relevant to this thesis, in a more formal setting. As already noted, we refer to, for instance, the books [12, 28, 30] or the survey papers [15, 76, 83] for more details.

4.1 A Proof Complexity Primer

We assume the existence of an infinite set *Vars* of boolean (or propositional logic) variables ranging over $\{0, 1\}$, where we identify 0 with *FALSE* and 1 with *TRUE*, respectively. We use the traditional set of logical connectives: negation $\neg$, conjunction $\wedge$, disjunction $\vee$, implication $\rightarrow$ and bi-implication (or equivalence) $\leftrightarrow$.

The set PROP of propositional logic formulas is the smallest set X such that

- $x \in X$ for all propositional logic variables $x \in \mathit{Vars}$,
- if $F, G \in X$ then $\bigl(F \wedge G\bigr), \bigl(F \vee G\bigr), \bigl(F \rightarrow G\bigr), \bigl(F \leftrightarrow G\bigr) \in X$,
- if $F \in X$ then $\bigl(\neg F\bigr) \in X$.

We write $\mathit{Vars}(F)$ to denote the set of variables of a formula F, i.e., $\mathit{Vars}(x) = \{x\}$, $\mathit{Vars}(\neg F) = \mathit{Vars}(F)$, $\mathit{Vars}(F \wedge G) = \mathit{Vars}(F) \cup \mathit{Vars}(G)$, and analogously for the other connectives.

Let α denote a truth value assignment, i.e., a function $\alpha : \mathit{Vars} \mapsto \{0, 1\}$. Then α is extended from variables to formulas in the canonical way by defining that $\alpha(\neg F) = 1$ if $\alpha(F) = 0$, $\alpha(F \wedge G) = 1$ if $\alpha(F) = \alpha(G) = 1$, $\alpha(F \vee G) = 1$ unless $\alpha(F) = \alpha(G) = 0$, $\alpha(F \rightarrow G) = 1$ unless $\alpha(F) = 1$ and $\alpha(G) = 0$, and $\alpha(F \leftrightarrow G) = 1$ if $\alpha(F) = \alpha(G)$. We say that F is

- *satisfiable* if there is an assignment α with $\alpha(F) = 1$,
- *valid* or *tautological* if all assignments satisfy F,

- *falsifiable* if there is an assignment α with $\alpha(F) = 0$,
- *unsatisfiable* or *contradictory* if all assignments falsify F.

If an assignment α satisfies a formula F, α is called a *model* of F. If α falsifies F, α is called a *counter-model*. The set of all tautological propositional logic formulas (or *tautologies*) F is denoted TAUTOLOGY. For more details, see [35] or any other standard textbook on logic.

The definition below from [12] is an adaption of the original definition in [33].

Definition 4.1 (Proof system). A *proof system* for a language L (or set L, depending on which terminology one prefers) is a polynomial-time algorithm P such that

1. for all $x \in L$ there is a string π (a *proof*) such that $P(x, \pi) = 1$,
2. for all $x \notin L$ and for all strings π it holds that $P(x, \pi) = 0$.

Note that P does not have to be polynomial-time in x only. If the proof π is large, P can use time polynomial in the size of the proof while checking it.

Let us define the *size* $S(x)$ of a string x to be the number of symbols in x. Then the *complexity* of a proof system P for a language L, which we denote *cplx*(P), is the smallest bounding function $g : \mathbb{N} \mapsto \mathbb{N}$ such that every $x \in L$ has a proof of size at most $g\bigl(S(x)\bigr)$, or in more formal notation

$$x \in L \Leftrightarrow \exists\pi \; S(\pi) \leq g\bigl(S(x)\bigr) \; \wedge \; P\bigl(x, \pi\bigr) = 1 \; . \tag{4.1}$$

If a proof system is of polynomial complexity, it is said to be *polynomially bounded* or *p-bounded*. Thus, NP is exactly the set of languages with polynomially bounded proof systems.

In this thesis, we are interested in proof systems for the set of all tautologies in propositional logic.

Definition 4.2 (Propositional proof system). A *propositional proof system* P is a proof system for TAUTOLOGY.

That is, a propositional proof system is a polynomial-time computable binary predicate P satisfying the following property: for all propositional logic formulas F it holds that $F \in$ TAUTOLOGY if and only if there exists a proof π of F such that $P\bigl(F, \pi\bigr)$ is true.

A quite common variation of this theme, a variation that we will focus on in the rest of this thesis, is to prove instead that formulas in conjunctive normal form (CNF formulas) are unsatisfiable. The reason that this is essentially the same problem is that it is possible to convert any propositional logic formula F to a CNF formula in such a way that it has only linearly larger size and is unsatisfiable if and only if the original formula is a tautology. One example of such a conversion is a transformation first used by Tseitin [81]. The idea in Tseitin's transformation is

$$
\begin{aligned}
G \doteq H_1 \wedge H_2 : \qquad & Tr\bigl(G\bigr) = \bigl(\neg x_G \vee x_{H_1}\bigr) \\
& \qquad \wedge \bigl(\neg x_G \vee x_{H_2}\bigr) \\
& \qquad \wedge \bigl(x_G \vee \neg x_{H_1} \vee \neg x_{H_2}\bigr)
\end{aligned}
$$

$$
\begin{aligned}
G \doteq H_1 \vee H_2 : \qquad & Tr\bigl(G\bigr) = \bigl(\neg x_G \vee x_{H_1} \vee x_{H_2}\bigr) \\
& \qquad \wedge \bigl(x_G \vee \neg x_{H_1}\bigr) \\
& \qquad \wedge \bigl(x_G \vee \neg x_{H_2}\bigr)
\end{aligned}
$$

$$
\begin{aligned}
G \doteq H_1 \rightarrow H_2 : \qquad & Tr\bigl(G\bigr) = \bigl(\neg x_G \vee \neg x_{H_1} \vee x_{H_2}\bigr) \\
& \qquad \wedge \bigl(x_G \vee x_{H_1}\bigr) \\
& \qquad \wedge \bigl(x_G \vee \neg x_{H_2}\bigr)
\end{aligned}
$$

$$
\begin{aligned}
G \doteq H_1 \leftrightarrow H_2 : \qquad & Tr\bigl(G\bigr) = \bigl(\neg x_G \vee \neg x_{H_1} \vee x_{H_2}\bigr) \\
& \qquad \wedge \bigl(\neg x_G \vee x_{H_1} \vee \neg x_{H_2}\bigr) \\
& \qquad \wedge \bigl(x_G \vee \neg x_{H_1} \vee \neg x_{H_2}\bigr) \\
& \qquad \wedge \bigl(x_G \vee x_{H_1} \vee x_{H_2}\bigr)
\end{aligned}
$$

Figure 4.1: Tseitin's transformation to CNF formulas.

to introduce a new variable x_G for each subformula $G \doteq H_1 \circ H_2$ in F, where we let $\circ$ denote one of the connectives $\wedge$, $\vee$, $\rightarrow$, or $\leftrightarrow$, and use $\doteq$ to denote syntactic equality. The formula F is then translated to conjunctive normal form by adding a set of clauses $Tr\bigl(G\bigr)$ for each subformula G which enforces that the the truth value of x_G is computed correctly given the truth values of x_{H_1} and x_{H_2}. These clauses $Tr\bigl(G\bigr)$ are presented in Figure 4.1. Finally, a unit clause $\neg x_F$ is added. It is easy to verify that the resulting CNF formula is unsatisfiable if and only if F is a tautology. In this way, any sound and complete system which produces refutations of formulas in conjunctive normal form can be considered as a general propositional proof system.

We have already argued that proving tautologies (or equivalently, as we have just seen, refuting unsatisfiable CNF formulas) is an important applied problem, but one other reason why proof complexity is interesting from a theoretical point of view is the following theorem.

Theorem 4.3 ([33]). *NP = co-NP if and only if there exists a polynomially bounded propositional proof system.*

Proof. ($\Rightarrow$) Obviously, TAUTOLOGY $\in$ $\mathsf{co\text{-}NP}$ since F is *not* a tautology if and only if $\neg F \in$ SATISFIABILITY. If $\mathsf{NP} = \mathsf{co\text{-}NP}$, then TAUTOLOGY $\in \mathsf{NP}$ has a polynomially bounded proof system by definition.

($\Leftarrow$) Conversely, assume that there exists a p-bounded propositional proof system. Then TAUTOLOGY $\in \mathsf{NP}$, and since TAUTOLOGY is complete for $\mathsf{co\text{-}NP}$ it follows that $\mathsf{NP} = \mathsf{co\text{-}NP}$. □

Since P is closed under complement, we have the following immediate corollary.

Corollary 4.4. *If all propositional proof systems have superpolynomial complexity, then* $\mathsf{P} \neq \mathsf{NP}$*.*

The conventional wisdom is that it should hold that $\mathsf{NP} \neq \mathsf{co\text{-}NP}$, but Corollary 4.4 explains why a proof of this still appears to be light years away. One line of research in proof complexity is to try to approach this distant goal by studying successively stronger propositional proof systems and relating their strengths. In this context, *polynomial simulations*, or *p-simulations*, play an important role.

Definition 4.5 (p-simulation). A propositional proof system P_1 *polynomially simulates*, or *p-simulates*, another propositional proof system P_2 if there exists a polynomial-time computable function f such that for all $F \in$ TAUTOLOGY it holds that $P_2\bigl(F, \pi\bigr) = 1$ if and only if $P_1\bigl(F, f(\pi)\bigr) = 1$.

If the complexity of two proof systems are within polynomial factors, we consider them to be "equally strong" for theoretical purposes.

Definition 4.6 (p-equivalence). Two propositional proof systems P_1 and P_2 are *polynomially equivalent*, or *p-equivalent*, if each proof system p-simulates the other.

Polynomial simulations define a partial order relation on proof systems. A natural question is whether there is a maximal element with respect to this ordering or not. This is not known, and there is little circumstantial evidence either way. Formally, let us say that a propositional proof system is *p-optimal* if it p-simulates every other propositional proof system. Then we have the following result.

Theorem 4.7 ([52]). *If* $\mathsf{EXP} = \mathsf{NEXP}$*, there is a p-optimal propositional proof system.*

This does not tell us too much, though, since this complexity class equality is considered implausible.

The definitions so far say nothing about how hard it might be to actually *find* proofs in the proof system P. Let us say that a *proof search algorithm* A_P for P is a deterministic algorithm A_P that takes as input a formula F and generates a proof π of F in the format specified by the proof system P (i.e., such that $P\bigl(F, \pi\bigr) = 1$) if F is valid and reports that F is falsifiable otherwise. Then the following definition from [12] captures a property that we would like our propositional proof system to have.

Definition 4.8 (Automatizability). Given a propositional proof system P and a function $f : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{N}$, we say that P is $f(n, S)$-*automatizable* if there exists a proof search algorithm A_P such that if $F \in$ TAUTOLOGY, then A_P on input F outputs a P-proof of F in time at most $f(n, S)$, where n is the size of F and S is the size of a smallest P-proof of F.

A proof system P is called *automatizable* if it is $f(n, S)$-automatizable for some $f(n, S) = \mathrm{poly}(n) \cdot \mathrm{poly}(S)$. The proof system P is *quasi-automatizable* if it is $f(n, S)$-automatizable for $f(n, S) = n^{c_1} \cdot \exp\bigl(\log^{c_2} S\bigr)$ for some constants c_1, c_2.

Note that automatizability seems to be the right definition because given a proof system P, this is in a sense the best we can hope for. If there are no small proofs of F in P to be found, then no proof search algorithm A_P in P can be expected to find proofs quickly. However, given a bound on the best any proof search algorithm for P can do, we want an algorithm A_P that performs well with respect to this bound.

Let us conclude this very brief introduction to proof complexity by giving examples of concrete propositional proof systems. No introduction to proof complexity can be complete without at least mentioning what a Frege system is. The next two definitions are (slightly adapted) from [28].

Definition 4.9 (Frege system). Let $F, F_1, \ldots, F_k$ be propositional logic formulas over the variables $x_1, \ldots, x_n$. A *Frege rule* is a pair

$$\bigl(\bigl\{F_1(x_1, \ldots, x_n), \ldots, F_k(x_1, \ldots, x_n)\bigr\}, F(x_1, \ldots, x_n)\bigr)$$

such that the implication $F_1(x_1, \ldots, x_n) \land \ldots \land F_k(x_1, \ldots, x_n) \rightarrow F(x_1, \ldots, x_n)$ is a tautology. Usually the rule is written as $\frac{F_1, \ldots, F_k}{F}$. A Frege rule with zero assumptions is called an *axiom schema.*

A Frege rule is applied by substituting arbitrary formulas for the variables $x_1, \ldots, x_n$. A *Frege proof* of a formula G is a sequence of formulas such that each formula follows from previous ones by an application of a Frege rule from a given set of rules and the last formula is G.

A *Frege system* $\mathfrak{F}$ is determined by a finite complete set of connectives B and a finite set of Frege rules such that $\mathfrak{F}$ is implicationally complete for the set of formulas over the basis B.

If Definition 4.9 seems very relaxed, it is because the details do not matter very much.

Theorem 4.10 ([33]). *Any two Frege systems p-simulate each other.*

Sadly, there are currently no strong lower bounds known for Frege systems (but see [27] for a survey of what is known). However, by restricting the model, somewhat similarly to what is done in circuit complexity, we get subsystems for which it is known how to prove superpolynomial lower bounds.

Definition 4.11 (Bounded-depth Frege system). Consider formulas in basis $\{\wedge, \vee, \neg\}$, The depth of a formula is the maximum number of alterations of connectives in it. A *depth-d Frege proof* is a Frege proof where all formulas in the proof sequence have depth at most d.

Theorem 4.12 ([51, 69]). *The pigeonhole principle formulas (encoding the statement that if $n + 1$ pigeons are placed in n pigeonholes, then at least one pigeonhole must contain more than one pigeon) require bounded-depth Frege proofs of size growing exponentially in n.*

Informally speaking, there seems to be an unfortunate trade-off for proof systems in that if a proof system is sufficiently powerful, then it is not automatizable. For instance, bounded-depth Frege systems are not automatizable under plausible cryptographic assumptions. More formally, we call $n \in \mathbb{N}$ a *Blum integer* if $n = pq$ for primes $p \equiv q \equiv 3 \pmod 4$. Then the following theorem is known.

Corollary 4.13 ([23]). *If factoring Blum integers is hard, then any proof system that can p-simulate bounded-depth Frege is not automatizable.*

The resolution proof systems, that we define next, can be viewed as a very limited form of a bounded-depth Frege system, namely depth-0 Frege. Even this proof system is likely not to be automatizable [6], but as was mentioned in Chapter 2 there are proof search algorithms for resolution that seem to work very well in practice.

4.2 Definition of the Resolution Proof System

A *literal* is either a propositional logic variable x or its negation, which we will from now on denote $\overline{x}$. Sometimes, though, it will be convenient to write x^1 for x and x^0 for $\overline{x}$. We define $\overline{\overline{x}} = x$. Two literals a and b are *strictly distinct* if $a \neq b$ and $a \neq \overline{b}$, i.e., if they refer to distinct variables.

A *clause* $C = a_1 \vee \cdots \vee a_k$ is a set of literals. Throughout this thesis, all clauses C are assumed to be nontrivial in the sense that all literals in C are pairwise strictly distinct (otherwise C is trivially true). We say that C is a *subclause* of D if $C \subseteq D$. A clause containing at most k literals is called a *k-clause*.

A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a set of clauses. A *k-CNF formula* is a CNF formula consisting of k-clauses. We define the *size* $S(F)$ of the formula F to be the total number of literals in F counted with repetitions. More often, we will be interested in the number of clauses $|F|$ of F.

In this thesis, when nothing else is stated it is assumed that A, B, C, D denote clauses, $\mathbb{C}, \mathbb{D}$ sets of clauses, x, y propositional variables, a, b, c literals, α, β truth value assignments and ν a truth value 0 or 1. We write

$$\alpha^{x=\nu}(y) = \begin{cases} \alpha(y) & \text{if } y \neq x, \\ \nu & \text{if } y = x, \end{cases} \tag{4.2}$$

$$\begin{aligned} F = \quad & (x \lor z) \land (\overline{z} \lor y) \land (x \lor \overline{y} \lor u) \land (\overline{y} \lor \overline{u}) \\ & \land\, (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w}) \end{aligned}$$

(a) CNF formula F.

1.	$x \lor z$	Axiom	9.	$x \lor y$	Res$(1, 2)$
2.	$\overline{z} \lor y$	Axiom	10.	$x \lor \overline{y}$	Res$(3, 4)$
3.	$x \lor \overline{y} \lor u$	Axiom	11.	$\overline{x} \lor u$	Res$(5, 6)$
4.	$\overline{y} \lor \overline{u}$	Axiom	12.	$\overline{x} \lor \overline{u}$	Res$(7, 8)$
5.	$u \lor v$	Axiom	13.	x	Res$(9, 10)$
6.	$\overline{x} \lor \overline{v}$	Axiom	14.	$\overline{x}$	Res$(11, 12)$
7.	$\overline{u} \lor w$	Axiom	15.	0	Res$(13, 14)$
8.	$\overline{x} \lor \overline{u} \lor \overline{w}$	Axiom			

(b) Resolution refutation of F.

Figure 4.2: Example resolution refutation.

to denote the truth value assignment that agrees with α everywhere except possibly at x, to which it assigns the value ν. We let $\mathit{Vars}(C)$ denote the set of variables and $\mathit{Lit}(C)$ the set of literals in a clause C.[1] This notation is extended to sets of clauses by taking unions. Also, we employ the standard notation $[n] = \{1, 2, \ldots, n\}$.

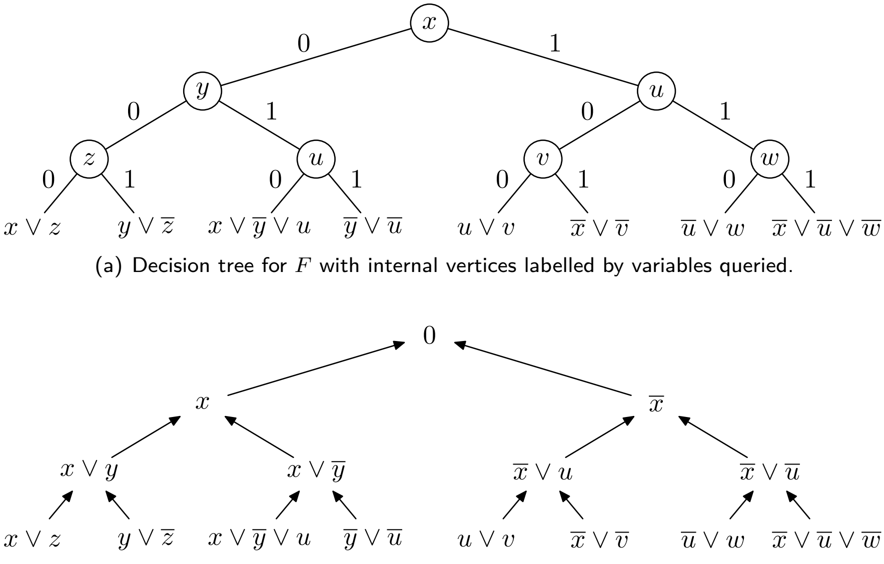
A *resolution derivation* $\pi : F \vdash A$ of a clause A from a CNF formula F is a sequence of clauses $\pi = \{D_1, \ldots, D_\tau\}$ such that $D_\tau = A$ and each line D_i, $i \in [\tau]$, either is one of the clauses in F (*axioms*) or is derived from clauses D_j, D_k in π with $j, k < i$ by the *resolution rule*

$$\frac{B \lor x \quad C \lor \overline{x}}{B \lor C} \; . \tag{4.3}$$

We refer to (4.3) as *resolution on the variable* x and to $B \lor C$ as the *resolvent* of $B \lor x$ and $C \lor \overline{x}$ on x. A *resolution refutation* π of a CNF formula F is a resolution derivation of the empty clause 0, i.e., the clause with no literals, from F. See Figure 4.2 for an example resolution refutation. Perhaps somewhat confusingly, π is sometimes also referred to as a *resolution proof* of F in the literature, since we can view F as being the encoding of the negation of a tautology as explained in Section 4.1. In this thesis, we will try to stick to talking about "refutations of F," but the terms "resolution refutation" and "resolution proof" in general will be used interchangeably.

For a formula F and a set of formulas $\mathcal{G} = \{G_1, \ldots, G_n\}$, we say that $\mathcal{G}$ *implies* F, denoted $\mathcal{G} \vDash F$, if every truth value assignment satisfying all formulas

[1] Although the notation $\mathit{Lit}(C)$ is slightly redundant given the definition of a clause as a set of literals, we include it for clarity.



(a) Decision tree for F with internal vertices labelled by variables queried.

(b) Corresponding resolution refutation of F.

Figure 4.3: Proof by example of implicational completeness of resolution.

$G \in \mathcal{G}$ must satisfy F as well. It is well known that resolution is sound and implicationally complete. That is, if there is a resolution derivation $\pi : F \vdash A$, then $F \vDash A$, and if $F \vDash A$, then there is a resolution derivation $\pi : F \vdash A'$ for some $A' \subseteq A$. In particular, F is unsatisfiable if and only if there is a resolution refutation of F.

We note that the soundness is not hard to argue—it follows from the fact that the resolution rule (4.3) is sound. Completeness is not immediately obvious, but let us sketch a proof. Given any unsatisfiable CNF formula F, we can build a decision tree for F, where we query some variable x in each vertex and branch left or right depending on the value assigned to x. Then the paths from the root downwards in the tree correspond to partial truth value assignments, and as soon as an assignment falsifies a clause, we add a leaf labelled by that clause. It is clear that we can build such a decision tree for any unsatisfiable formula F, and if we then turn this decision tree upside down, we have (essentially) a resolution refutation of F. Figure 4.3 gives a proof by example of this fact, and although we omit the details it is not hard to make this into a formal proof.

With every resolution derivation $\pi : F \vdash A$ we can associate a DAG G_π, with the clauses in π labelling the vertices and with edges from the assumption clauses to the resolvent for each application of the resolution rule (4.3). There might be several different derivations of a clause C in π, but if so we can label each occurrence of C with a time-stamp when it was derived and keep track of which copy of C is used where. A resolution derivation π is *tree-like* if any clause in the derivation is used at most once as a premise in an application of the resolution rule, i.e., if G_π is a tree. (We may make different "time-stamped" vertex copies of the axiom clauses in order to make G_π into a tree). As we can see from Figure 4.3(b), our example refutation in Figure 4.2 is tree-like.

Given this definition of the resolution proof system, we can define the *length* $L(\pi)$ of a resolution derivation π as the number of clauses in it, and the *width* $W(\pi)$ of a derivation is the size of a largest clause in it. For instance, the refutation in Figure 4.2 has length 15 and width 3. However, in order to define space in a natural way and to be able to reason about trade-offs between measures, it is convenient to describe resolution is a slightly different way.

Following the exposition in [39], a resolution proof can be seen as a Turing machine computation, with a special read-only input tape from which the axioms can be downloaded and a working memory where all derivation steps are made. Then the *clause space* of a resolution proof is the maximum number of clauses that need to be kept in memory simultaneously during a verification of the proof. The *variable space* is the maximum total space needed, where also the width of the clauses is taken into account. The formal definitions follow.

Definition 4.14 (Resolution ([4])). A *clause configuration* $\mathbb{C}$ is a set of clauses. A sequence of clause configurations $\{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ is a *resolution derivation* from a CNF formula F if $\mathbb{C}_0 = \emptyset$ and for all $t \in [\tau]$, $\mathbb{C}_t$ is obtained from $\mathbb{C}_{t-1}$ by one[2] of the following rules:

Axiom Download $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C\}$ for some $C \in F$.

Erasure $\mathbb{C}_t = \mathbb{C}_{t-1} \setminus \{C\}$ for some $C \in \mathbb{C}_{t-1}$.

Inference $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{D\}$ for some D inferred by resolution from $C_1, C_2 \in \mathbb{C}_{t-1}$.

A resolution derivation $\pi : F \vdash A$ of a clause A from a formula F is a derivation $\{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ such that $\mathbb{C}_\tau = \{A\}$. A *resolution refutation* of F is a derivation of the empty clause 0 from F.

Definition 4.15 (Length, width, space). The *width* $W(C)$ of a clause C is $|C|$, i.e., the number of literals in it. The width of a clause configuration $\mathbb{C}$ is $W(\mathbb{C}) = \max_{C \in \mathbb{C}}\{W(C)\}$. The *clause space* of a configuration $\mathbb{C}$ is $Sp(\mathbb{C}) = |\mathbb{C}|$, i.e., the number of clauses in $\mathbb{C}$, and the *variable space* is $VarSp(\mathbb{C}) = \sum_{C \in \mathbb{C}} W(C)$.

Let π be a resolution derivation. Then:

- The *length* $L(\pi)$ of π is the number of axiom download and inference steps in π.
- The *width* of π is $W(\pi) = \max_{\mathbb{C} \in \pi}\{W(\mathbb{C})\}$.

[2] In some previous papers, resolution is defined so as to allow every derivation step to *combine* one or zero applications of each of the three derivation rules. Therefore, some of the bounds stated in this thesis for space as defined next are off by a constant as compared to the cited sources.

- The *clause space* of π is $Sp(\pi) = \max_{\mathbb{C}\in\pi}\{Sp(\mathbb{C})\}$.
- The *variable space* of π is $\mathit{VarSp}(\pi) = \max_{\mathbb{C}\in\pi}\{\mathit{VarSp}(\mathbb{C})\}$.

We define the length of deriving a clause A from F as $L(F \vdash A) = \min_{\pi : F \vdash A}\{L(\pi)\}$, where the minimum is taken over all resolution derivations of A. The width $W(F \vdash A)$, clause space $Sp(F \vdash A)$, and variable space $\mathit{VarSp}(F \vdash A)$ of deriving A from F are defined completely analogously. The length, width, clause space and variable space of refuting F is $L(F \vdash 0)$, $W(F \vdash 0)$, $Sp(F \vdash 0)$, and $\mathit{VarSp}(F \vdash 0)$, respectively, where as before 0 denotes the contradictory empty clause.

In this thesis, we will be almost exclusively interested in the clause space of general resolution refutations. When we write simply "space" for brevity, we mean clause space.

As an aside, we note that if one wanted to be really precise, the size and space measures should probably measure the number of *bits* needed rather than the number of literals. However, counting literals makes matters substantially cleaner, and the difference is at most a logarithmic factor. Therefore, counting literals seems to be the established way of measuring formula size and variable space.

Using the "configuration-style" description of resolution in Definition 4.14, a tree-like resolution derivation can be defined as a derivation where a clause has to be erased as soon as it has been used in an inference step. Restricting the resolution derivations to tree-like resolution, we can define the minimum length $L_{\mathfrak{T}}(F \vdash 0)$, clause space $Sp_{\mathfrak{T}}(F \vdash 0)$, and variable space $\mathit{VarSp}_{\mathfrak{T}}(F \vdash 0)$ of refuting F in tree-like resolution in analogy with the measures in Definition 4.15. Note that the minimum width measures in general and tree-like resolution coincide, so it makes no sense to make a separate definition for $W_{\mathfrak{T}}(F \vdash 0)$.

For technical reasons, it is sometimes convenient to add a rule for *weakening* in resolution, saying that we can always derive a weaker clause $C' \supseteq C$ from C. It is easy to show that any weakening steps can always be eliminated from a resolution refutation without increasing the length, width or space.

Restrictions are another technical tool that we will use to to simplify some of the proofs.

Definition 4.16 (Restriction). A *partial assignment* or *restriction* ρ is a partial function $\rho : X \mapsto \{0, 1\}$, where X is a set of Boolean variables. We identify ρ with the set of literals $\{a_1, \ldots, a_m\}$ set to true by ρ. The *ρ-restriction* of a clause C is defined to be

$$C\restriction_{\rho} = \begin{cases} 1 & \text{(i.e., the trivially true clause) if } \mathit{Lit}\bigl(C\bigr) \cap \rho \neq \emptyset, \\ C \setminus \{\overline{a} \mid a \in \rho\} & \text{otherwise.} \end{cases}$$

This definition is extended to set of clauses by taking unions.

We write $\rho(\neg C)$ to denote the minimal restriction fixing C to false, i.e., $\rho(\neg C) = \{\overline{a} \mid a \in C\}$.

$\pi =$

1.	$x \lor z$	Axiom in F
2.	$\overline{z} \lor y$	Axiom in F
3.	$x \lor \overline{y} \lor u$	Axiom in F
4.	$\overline{y} \lor \overline{u}$	Axiom in F
5.	$u \lor v$	Axiom in F
6.	$\overline{x} \lor \overline{v}$	Axiom in F
7.	$\overline{u} \lor w$	Axiom in F
8.	$\overline{x} \lor \overline{u} \lor \overline{w}$	Axiom in F
9.	$x \lor y$	$\mathsf{Res}(1,2)$
10.	$x \lor \overline{y}$	$\mathsf{Res}(3,4)$
11.	$\overline{x} \lor u$	$\mathsf{Res}(5,6)$
12.	$\overline{x} \lor \overline{u}$	$\mathsf{Res}(7,8)$
13.	x	$\mathsf{Res}(9,10)$
14.	$\overline{x}$	$\mathsf{Res}(11,12)$
15.	0	$\mathsf{Res}(13,14)$

(a) Resolution refutation π.

$\pi{\restriction}_x =$

1.	1	
2.	$\overline{z} \lor y$	Axiom in $F{\restriction}_x$
3.	1	
4.	$\overline{y} \lor \overline{u}$	Axiom in $F{\restriction}_x$
5.	$u \lor v$	Axiom in $F{\restriction}_x$
6.	$\overline{v}$	Axiom in $F{\restriction}_x$
7.	$\overline{u} \lor w$	Axiom in $F{\restriction}_x$
8.	$\overline{u} \lor \overline{w}$	Axiom in $F{\restriction}_x$
9.	1	
10.	1	
11.	u	$\mathsf{Res}(5,6)$
12.	$\overline{u}$	$\mathsf{Res}(7,8)$
13.	1	
14.	0	$\mathsf{Res}(11,12)$
15.	0	

(b) Restriction $\pi{\restriction}_x$ setting x to true.

Figure 4.4: Proof by example that restrictions preserve resolution refutations.

Proposition 4.17. *If π is a resolution refutation of F and ρ is a restriction on $\mathit{Vars}(F)$, then $\pi{\restriction}_\rho$ can be transformed into a resolution refutation of $F{\restriction}_\rho$ in at most the same length, width and space as π.*

See Figure 4.4 for an illustration of this using our running example resolution refutation. In this case, the restriction results in a legal resolution refutation, but in general we might need the weakening rule to show that $\pi{\restriction}_\rho$ is a refutation of $F{\restriction}_\rho$. The formal proof is an easy induction over the derivations steps in π.

4.3 A Review of Some Results

It is not hard to show that any unsatisfiable CNF formula F over n variables is refutable in length $2^{n+1} - 1$, using the decision tree construction sketched in Figure 4.3. Also, the maximal refutation width is clearly at most the number of variables $n + 1$. Esteban and Torán [39] proved that the clause space of refuting F is upper-bounded by the formula size. More precisely, the minimal clause space is at most the number of clauses, or the number of variables, plus a small constant, or in formal notation $\mathit{Sp}(F \vdash 0) \leq \min\{|F|, |\mathit{Vars}(F)|\} + \mathrm{O}(1)$. Again, this follows by studying resolution refutations constructed as in Figure 4.3. The height of the decision tree is at most the number of variables, and it can be shown that any resolution refutation described by a binary tree of height at most h can be carried out in clause space $h + \mathrm{O}(1)$.

We will need the fact that there are polynomial-size families of k-CNF formulas that are very hard with respect to length, width and clause space, essentially meeting the upper bounds just stated.

Theorem 4.18 ([4, 13, 18, 21, 29, 79, 82]). *There are arbitrarily large unsatisfiable* 3*-CNF formulas* F_n *of size* $\Theta(n)$ *with* $\Theta(n)$ *clauses and* $\Theta(n)$ *variables for which it holds that* $L(F_n \vdash 0) = \exp(\Theta(n))$*,* $W(F_n \vdash 0) = \Theta(n)$ *and* $Sp(F_n \vdash 0) = \Theta(n)$*.*

Clearly, for such formulas F_n it must also hold that $\Omega(n) = \mathit{VarSp}(F_n \vdash 0) = \mathrm{O}\bigl(n^2\bigr)$. We note in passing that determining the exact variable space complexity of a formula family as in Theorem 4.18, or even proving a lower bound $\omega(n)$ on the variable space, was mentioned as an open problem in [4]. To the best of our knowledge, this problem is still unsolved.

If a resolution refutation has constant width, it is easy to see that it can be carried out in length polynomial in the number of variables (just count the maximum possible number of distinct clauses). Conversely, if all refutations of a formula are very wide, it seems reasonable that any refutation of this formula must be very long as well. This intuition was made precise by Ben-Sasson and Wigderson [21]. We state their theorem in the more explicit form of Segerlind [76].

Theorem 4.19 ([21]). *The width of refuting an unsatisfiable CNF formula* F *is bounded from above by*

$$W(F \vdash 0) \leq W(F) + 1 + 3\sqrt{n \ln L(F \vdash 0)} \enspace ,$$

where n *is the number of variables in* F*.*

Bonet and Galesi [25] showed that this bound on width in terms of length is essentially optimal. For the special case of tree-like resolution, however, it is possible get rid of the dependence of the number of variables and obtain a tighter bound.

Theorem 4.20 ([21]). *The width of refuting an unsatisfiable CNF formula* F *in tree-like resolution is bounded from above by* $W(F \vdash 0) \leq W(F) + \log L_{\mathfrak{T}}(F \vdash 0)$*.*

For reference, we collect the result in [25] together with some other bounds showing that there are formulas that are easy with respect to length but moderately hard with respect to width and clause space, and state them as a theorem.[3]

Theorem 4.21 ([4, 25, 78]). *There are arbitrarily large unsatisfiable* 3*-CNF formulas* F_n *of size* $\Theta\bigl(n^3\bigr)$ *with* $\Theta\bigl(n^3\bigr)$ *clauses and* $\Theta\bigl(n^2\bigr)$ *variables such that* $W(F_n \vdash 0) = \Theta(n)$ *and* $Sp(F_n \vdash 0) = \Theta(n)$*, but for which there are resolution refutations* $\pi_n : F_n \vdash 0$ *in length* $L(\pi_n) = \mathrm{O}\bigl(n^3\bigr)$*, width* $W(\pi_n) = \mathrm{O}(n)$ *and clause space* $Sp(\pi_n) = \mathrm{O}(n)$*.*

[3] Note that [25], where an explicit resolution refutation upper-bounding the proof complexity measures is presented, does not talk about clause space, but it is straightforward to verify that the refutation there can be carried out in length $\mathrm{O}\bigl(n^3\bigr)$ and clause space $\mathrm{O}(n)$.

As was mentioned in Chapter 2, the fact that all known lower bounds on refutation clause space coincided with lower bounds on width lead to the conjecture that the width measure is a lower bound for the clause space measure. This conjecture was proven true by Atserias and Dalmau [10].

Theorem 4.22 ([10]). *For any unsatisfiable CNF formula F, $Sp(F \vdash 0) - 3 \geq W(F \vdash 0) - W(F)$.*

In other words, the extra clause space exceeding the minimum 3 needed for any resolution refutation is bounded from below by the extra width exceeding the width of the formula.

An immediate corollary of Theorem 4.22 is that for polynomial-size k-CNF formulas, constant clause space implies polynomial proof length. We are interested in finding out what holds in the other direction, i.e., if upper bounds on length imply upper bounds on space. For the special case of tree-like resolution, it is known that there is an upper bound on clause space in terms of length exactly analogous to the one on width in terms of length in Theorem 4.20.

Theorem 4.23 ([39]). *The clause space of refuting an unsatisfiable CNF formula F in tree-like resolution is bounded from above by $Sp_{\mathfrak{T}}(F \vdash 0) \leq \lceil \log L_{\mathfrak{T}}(F \vdash 0) \rceil + 2$.*

For general resolution, since clause space is lower-bounded by width according to Theorem 4.22, the separation of width and length of [25] in Theorem 4.21 tells us that k-CNF formulas refutable in polynomial length can still have “somewhat spacious” minimum-space refutations. But exactly how spacious can they be? Does space behave as width with respect to length also in general resolution, or can one get stronger lower bounds on space for formulas refutable in polynomial length?

All polynomial lower bounds on clause space known prior to this thesis can be explained as immediate consequences of Theorem 4.22 applied on lower bounds on width. Clearly, any space lower bounds derived in this way cannot get us beyond the “Ben-Sasson–Wigderson barrier” implied by Theorem 4.19 saying that if the width of refuting F is $\omega\bigl(\sqrt{|F| \log|F|}\bigr)$, then the length of refuting F must be superpolynomial in $|F|$. Also, since matching upper bounds on clause space have been known for all of these formula families, they have not been candidates for showing stronger separations of space and length. Thus, the best known separation of clause space and length prior to this thesis was provided by the formulas in Theorem 4.21 refutable in linear length $L(F_n \vdash 0) = \mathrm{O}(|F_n|)$ but requiring space $Sp(F_n \vdash 0) = \Theta\bigl(\sqrt[3]{|F_n|}\bigr)$, as implied by the same bound on width.

Let us also discuss upper bounds on what kind of separations are a priori possible. Given any resolution refutation $\pi : F \vdash 0$, we can write down its DAG representation G_π (described on page 42) with $L(\pi)$ vertices corresponding to the clauses, and with all non-source vertices having fan-in 2. We can then transform π into as space-efficient a refutation as possible by considering an optimal black pebbling of G_π (soon to be formally defined in Definition 5.1) as follows: when a pebble is placed on a vertex we derive the corresponding clause, and when the pebble is removed again we erase the clause from memory. This yields a refutation π' in clause space $\mathsf{Peb}(G_\pi)$ (incidentally, this is the original definition in [39] of the clause space of a resolution refution π). Since it is known that any constant indegree DAG on n vertices can be black-pebbled in cost $\mathrm{O}(n/\log n)$ (see Theorem 5.4), this shows that $\mathit{Sp}(F \vdash 0) = \mathrm{O}\bigl(L(F \vdash 0)/\log L(F \vdash 0)\bigr)$ is an upper bound on space in terms of length.

Now we can rephrase the question above about space and length in the following way: Is there a Ben-Sasson–Wigderson kind of lower bound, say $L(F \vdash 0) = \exp\bigl(\Omega\bigl(\mathit{Sp}(F \vdash 0)^2/|F|\bigr)\bigr)$, on length in terms of space? Or do there exist k-CNF formulas F with short refutations but maximum possible refutation space $\mathit{Sp}(F \vdash 0) = \Omega\bigl(L(F \vdash 0)/\log L(F \vdash 0)\bigr)$ in terms of length? Note that the refutation length $L(F \vdash 0)$ must indeed be short in this case—essentially linear, since any formula F can be refuted in space $\mathrm{O}(|F|)$ as was noted above. Or is the relation between refutation space and refutation length somewhere in between these extremes?

This is the main question addressed in this thesis. We show that clause space and length can be strongly separated in the sense that there are formula families with maximum possible refutation clause space in terms of length. The same result also yields an almost optimal separation of clause space and width.

Theorem 4.24 (Corollary 2.6 restated). *For all $k \geq 6$ there is a family $\{F_n\}_{n=1}^{\infty}$ of k-CNF formulas of size $\Theta(n)$ that can be refuted in resolution in length $L(F_n \vdash 0) = \mathrm{O}(n)$ and width $W(F_n \vdash 0) = \mathrm{O}(1)$ but require clause space $\mathit{Sp}(F_n \vdash 0) = \Omega(n/\log n)$.*

Chapter 5

Pebble Games and Pebbling Contradictions

Pebble games were originally devised for studying programming languages and compiler construction, but have later found a variety of applications in computational complexity theory. In connection with resolution, pebble games have been employed both to analyze resolution derivations with respect to how much memory they consume (using the original definition of space in [39]) and to construct CNF formulas which are hard for different restricted variants of resolution in various respects (see for example [5, 19, 24, 26]). An excellent survey of pebbling up to ca. 1980 is [68].

This chapter presents formal definitions of pebble games and pebbling contradictions, and gives precise statements of some previously known results relevant to this thesis. At the end of Section 5.2, we also try to provide some of the intuition behind the proofs of our results.

5.1 Pebble Games

The black pebbling price of a DAG G captures the memory space, i.e., the number of registers, required to perform the deterministic computation described by G. The space of a non-deterministic computation is measured by the black-white pebbling price of G. We say that vertices of G with indegree 0 are *sources* and that vertices with outdegree 0 are *sinks* or *targets*. In the following, unless otherwise stated we will assume that all DAGs under discussion have a unique sink, and this sink will always be denoted z. The next definition is adapted from [34], though we use the established pebbling terminology introduced by [46].

Definition 5.1 (Pebble game). Suppose that G is a DAG with sources S and a unique target z. The *black-white pebble game* on G is the following one-player game. At any point in the game, there are black and white pebbles placed on some vertices of G, at most one pebble per vertex. A *pebble configuration* is a pair of subsets $\mathbb{P} = (B, W)$ of $V(G)$, comprising the black-pebbled vertices B and white-pebbled vertices W. The rules of the game are as follows:

1. If all immediate predecessors of an empty vertex v have pebbles on them, a black pebble may be placed on v. In particular, a black pebble can always be placed on any vertex in S.
2. A black pebble may be removed from any vertex at any time.
3. A white pebble may be placed on any empty vertex at any time.
4. If all immediate predecessors of a white-pebbled vertex v have pebbles on them, the white pebble on v may be removed. In particular, a white pebble can always be removed from a source vertex.

A *black-white pebbling* from (B_1, W_1) to (B_2, W_2) in G is a sequence of pebble configurations $\mathcal{P} = \{\mathbb{P}_0, \ldots, \mathbb{P}_\tau\}$ such that $\mathbb{P}_0 = (B_1, W_1)$, $\mathbb{P}_\tau = (B_2, W_2)$, and for all $t \in [\tau]$, $\mathbb{P}_t$ follows from $\mathbb{P}_{t-1}$ by one of the rules above. If $(B_1, W_1) = (\emptyset, \emptyset)$, we say that the pebbling is *unconditional*, otherwise it is *conditional*.

The *cost* of a pebble configuration $\mathbb{P} = (B, W)$ is $\mathit{cost}(\mathbb{P}) = |B \cup W|$ and the cost of a pebbling $\mathcal{P} = \{\mathbb{P}_0, \ldots, \mathbb{P}_\tau\}$ is $\max_{0 \le t \le \tau}\{\mathit{cost}(\mathbb{P}_t)\}$. The *black-white pebbling price* of (B, W), denoted $\mathit{BW\text{-}Peb}(B, W)$, is the minimum cost of any unconditional pebbling reaching (B, W).

A *complete pebbling* of G, also called a *pebbling strategy* for G, is an unconditional pebbling reaching $(\{z\}, \emptyset)$. The *black-white pebbling price* of G, denoted $\mathit{BW\text{-}Peb}(G)$, is the minimum cost of any complete black-white pebbling of G.

A *black pebbling* is a pebbling using black pebbles only, i.e., having $W_t = \emptyset$ for all t. The *(black) pebbling price* of G, denoted $\mathit{Peb}(G)$, is the minimum cost of any complete black pebbling of G.

See Figure 5.1 for an example of a complete black-white pebbling.

We think of the moves in a pebbling as occurring at integral time intervals $t = 1, 2, \ldots$ and talk about the pebbling move "at time t" (which is the move resulting in configuration $\mathbb{P}_t$) or the moves "during the time interval $[t_1, t_2]$".

The only pebblings we are really interested in are complete pebblings of G. However, when we prove lower bounds for pebbling price it will sometimes be convenient to be able to reason in terms of partial pebbling move sequences, i.e., conditional pebblings. One can think of conditional pebblings as pebblings that receive the start configuration (B_1, W_1) "as a gift", and are also allowed to leave (B_2, W_2) without "cleaning up" when they finish. It is clear that we can assume that $(B_1, W_1) = (B_1, \emptyset)$ and $(B_2, W_2) = (\emptyset, W_2)$ since we can freely place white pebbles on G and freely remove black pebbles. The way the gift can help us is that we get black pebbles at the beginning for free, and are allowed to leave white pebbles without having to do the hard pebbling work of removing them.

There is an extensive literature on pebbling, mostly from the 70s and 80s. We just quickly mention four results relevant to this thesis.

Perhaps the simplest graphs to pebble are complete binary trees T_h of height h. The black pebbling price of T_h can be established by an easy induction over the tree height. For black-white pebbling, general bounds for the pebbling price of trees

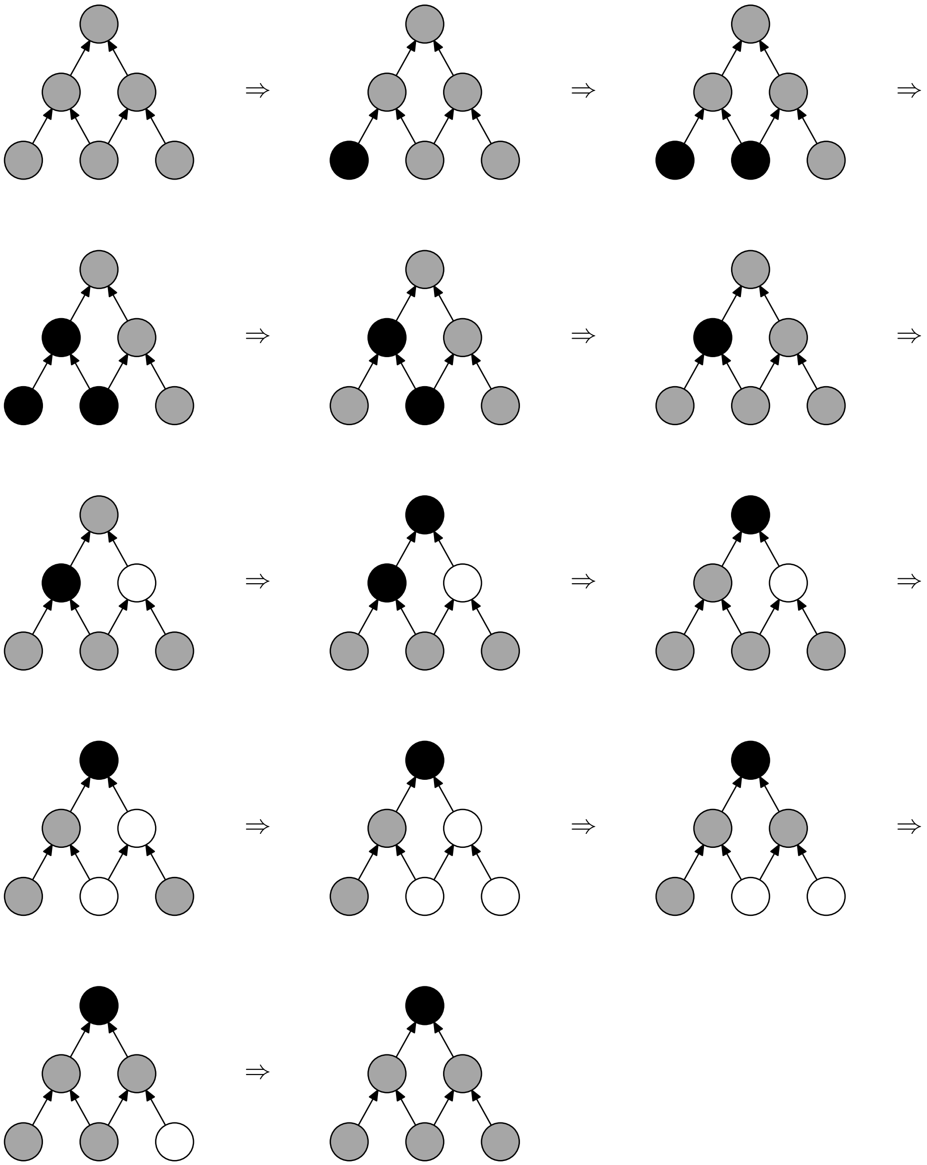


Figure 5.1: Complete black-white pebbling of pyramid of height 2.

of any arity were presented in [55] For the case of binary trees, this result can be simplified to the exact equality below, which is proven in Section 6.2.

Theorem 5.2. *For the complete binary tree T_h of height $h \geq 1$ it holds that $\mathsf{Peb}(T_h) = h + 2$ and $\mathsf{BW\text{-}Peb}(T_h) = \lfloor \frac{h}{2} \rfloor + 3$.*

Another class of DAGs that we will study are so-called pyramid graphs. We have not defined pyramid graphs formally yet (this will be done in Definition 6.4) but hopefully it is clear from the example pyramid of height 2 used a number of times already in this thesis (most recently in Figure 5.1) what pyramids look like.

Theorem 5.3 ([32, 49]). *For the pyramid graph Π_h of height $h \geq 1$ it holds that $\mathsf{Peb}(\Pi_h) = h + 2$ and $\mathsf{BW\text{-}Peb}(\Pi_h) = h/2 + \mathrm{O}(1)$.*

In general, we are interested in DAGs with as high a pebbling price as possible measured in terms of the number of vertices. For a DAG G with n vertices and constant in-degree, the best we can hope for is $\mathrm{O}(n/\log n)$.

Theorem 5.4 ([46]). *For directed acyclic graphs G with n vertices and constant maximum indegree, it holds that $\mathsf{Peb}(G) = \mathrm{O}\big(n/\log n\big)$.*

This bound is asymptotically tight for black-white pebbling, and thus for black pebbling as well.[1]

Theorem 5.5 ([42, 67]). *There is a family $\{G_n\}_{n=1}^{\infty}$ of explicitly constructible DAGs with $\Theta(n)$ vertices and vertex indegree 2 for all non-source vertices such that $\mathsf{BW\text{-}Peb}(G) = \Omega(n/\log n)$.*

We remark that no explicit constructions were known at the time of the original theorems in [42, 67]. What is needed is explicitly constructible superconcentrators of linear density, and it has since been shown how to build such graphs (with [7] currently being the best construction as far as we are aware).

It should be pointed out that although the black and black-white pebbling prices coincide asymptotically in all of the theorems above, this is not the case in general. In [47], a family of DAGs with a quadratic difference in the number of pebbles between the black and the black-white pebble game was presented. We note that this is the best separation possible, since by [56] the difference in black and black-white pebbling price can be at most quadratic.

5.2 Pebbling Contradictions

A *pebbling contradiction* defined on a DAG G is a CNF formula that encodes the pebble game on G by postulating the sources to be true and the target to be false, and specifying that truth propagates through the graph according to the pebbling rules. The definition below is a generalization of formulas previously studied in [24, 70].

[1] We note that in several papers, the result in Theorem 5.5 is incorrectly attributed to [54], but [54] itself gives the correct reference.

Definition 5.6 (Pebbling contradiction [21]). Suppose that G is a DAG with sources S, a unique target z and with all non-source vertices having indegree 2, and let $d > 0$ be an integer. Associate d distinct variables $x(v)_1, \ldots, x(v)_d$ with every vertex $v \in V(G)$. The dth degree *pebbling contradiction* over G, denoted Peb^d_G, is the conjunction of the following clauses:

- $\bigvee_{i=1}^{d} x(s)_i$ for all $s \in S$ (*source axioms*),
- $\overline{x(u)}_i \lor \overline{x(v)}_j \lor \bigvee_{l=1}^{d} x(w)_l$ for all $i, j \in [d]$ and all $w \in V(G) \setminus S$, where u, v are the two predecessors of w (*pebbling axioms*),
- $\overline{x(z)}_i$ for all $i \in [d]$ (*target* or *sink axioms*).

The formula Peb^d_G is a (2+d)-CNF formula with $\mathrm{O}\bigl(d^2 \cdot |V(G)|\bigr)$ clauses over $d \cdot |V(G)|$ variables. An example pebbling contradiction is presented in Figure 3.2 on page 27.

Although any constant indegree will be fine for the results covered in this subsection, we restrict our attention to DAGs with vertex indegree 2 for all non-source vertices since such are the graphs that will be studied in the rest of this thesis.

It was observed in [19] that Peb^d_G can be refuted in resolution by deriving $\bigvee_{i=1}^{d} x(v)_i$ for all $v \in V(G)$ inductively in topological order and then resolving with the target axioms $\overline{x(z)}_i$, $i \in [d]$. Writing down this resolution proof, one gets the following proposition (which is proven together with Proposition 5.10 below).

Proposition 5.7 ([19]). *For any DAG G with all vertices having indegree 0 or 2, there is a resolution refutation $\pi : \mathit{Peb}^d_G \vdash 0$ in length $L(\pi) = \mathrm{O}\bigl(d^2 \cdot |V(G)|\bigr)$ and width $W(\pi) = \mathrm{O}(d)$.*

Tree-like resolution is good at refuting first-degree pebbling contradictions Peb^1_G but is bad at refuting Peb^d_G for $d \geq 2$.

Theorem 5.8 ([16]). *For any DAG G with all vertices having indegree 0 or 2, there is a tree-like resolution refutation π of Peb^1_G such that $L(\pi) = \mathrm{O}(|V(G)|)$ and $\mathit{Sp}(\pi) = \mathrm{O}(1)$.*

Theorem 5.9 ([19]). *For any DAG G with all vertices having indegree 0 or 2, $L_{\mathfrak{T}}(\mathit{Peb}^2_G \vdash 0) = 2^{\Omega(\mathsf{Peb}(G))}$.*

As to space, it is not too difficult to see that the black pebbling price of G provides an upper bound for the refutation clause space of Peb^d_G.

Proposition 5.10. *For any DAG G with vertex indegrees 0 or 2, $\mathit{Sp}(\mathit{Peb}^d_G \vdash 0) \leq \mathsf{Peb}(G) + \mathrm{O}(1)$.*

Essentially, this is just a matter of combining an optimal black pebbling of G with the resolution refutation idea from [19] sketched above. Since we need the upper bounds on width and space in Propositions 5.7 and 5.10 in the proof of our results, we write down the details for completeness.

Proof of Propositions 5.7 and 5.10. Consider first the bound on space. Given a black pebbling of G, we construct a resolution refutation of Peb^d_G such that if at some point in time there are black pebbles on a set of vertices V, then we have the clauses $\bigl\{\bigvee_{i=1}^{d} x(v)_i \mid v \in V\bigr\}$ in memory. When some new vertex v is pebbled, we derive $\bigvee_{i=1}^{d} x(v)_i$ from the clauses already in memory. We claim that with a little care, this can be done in constant extra space independent of d. When a black pebble is removed from v, we erase the clause $\bigvee_{i=1}^{d} x(v)_i$. We conclude the resolution proof by resolving $\bigvee_{i=1}^{d} x(z)_i$ for the target z with all target axioms $\overline{x(z)}_i$, $i \in [d]$, in space 3.

It is clear that given our claim about the constant extra space needed when a vertex is black-pebbled, this yields a resolution refutation in space equal to the pebbling cost plus some constant. In particular, given an optimal black pebbling of G, we get a refutation in space $\mathsf{Peb}(G) + \mathrm{O}(1)$.

To prove the claim, note first that it trivially holds for source vertices v, since $\bigvee_{i=1}^{d} x(v)_i$ is an axiom of the formula. Suppose for a non-source vertex r with predecessors p and q that at some point in time a black pebble is placed on r. Then p and q must be black-pebbled, so by induction we have the clauses $\bigvee_{i=1}^{d} x(p)_i$ and $\bigvee_{j=1}^{d} x(q)_j$ in memory. We will use that the clause $\overline{x(p)}_i \lor \bigvee_{l=1}^{d} x(r)_l$ for any i can be derived in additional space 3 by resolving $\bigvee_{j=1}^{d} x(q)_j$ with $\overline{x(p)}_i \lor \overline{x(q)}_j \lor \bigvee_{l=1}^{d} x(r)_l$ for $j \in [d]$, leaving the easy verification of this fact to the reader. To derive $\bigvee_{l=1}^{d} x(r)_l$, first resolve $\bigvee_{i=1}^{d} x(p)_i$ with $\overline{x(p)}_1 \lor \bigvee_{l=1}^{d} x(r)_l$ to get $\bigvee_{i=2}^{d} x(p)_i \lor \bigvee_{l=1}^{d} x(r)_l$, and then resolve this clause with the clauses $\overline{x(p)}_i \lor \bigvee_{l=1}^{d} x(r)_l$ for $i = 2, \ldots, d$ one by one to get $\bigvee_{l=1}^{d} x(r)_l$ in total extra space 4.

It is easy to see that this proof has width $\mathrm{O}(d)$, which proves the claim about width in Proposition 5.7. To get the claim about length, we observe that the subderivation needed when a vertex is black-pebbled has length $\mathrm{O}\bigl(d^2\bigr)$. If we use a pebbling that black-pebbles all vertices once in topological order without ever removing a pebble, we get a refutation in length $L(\pi) = \mathrm{O}\bigl(d^2 \cdot |V(G)|\bigr)$. □

Thus, the refutation clause space of a pebbling contradiction is upper-bounded by the black pebbling price of the underlying DAG. Proposition 5.10 is not quite an optimal strategy with respect to clause space, though. For binary trees, [40] improved this bound somewhat to $Sp(Peb^2_{T_h} \vdash 0) \leq \frac{2}{3}h + \mathrm{O}(1)$ by constructing resolution proofs that try to mimic not black pebblings but instead optimal *black-white* pebblings of T_h as presented in [55]. And for one variable per vertex, we know from Theorem 5.8 that $Sp(Peb^1_G \vdash 0) = \mathrm{O}(1)$.

Proving lower bounds on space for pebbling contradictions of degree $d \geq 2$ has turned out to be much harder. For quite some time there was no lower bound on $Sp(Peb^d_G \vdash 0)$ for any DAG G in general resolution (in terms of pebbling price or otherwise). In [40], a lower bound $Sp_{\mathfrak{T}}(Peb^d_{T_h} \vdash 0) = h + \mathrm{O}(1)$ was obtained for the special case of tree-like resolution. Unfortunately, this does not tell us anything about general resolution. For tree-like resolution, if the only way of deriving a clause D is from clauses C_1, C_2 such that $Sp_{\mathfrak{T}}(F \vdash C_i) \geq s$, then it holds that $Sp_{\mathfrak{T}}(F \vdash D) \geq s + 1$ since one of the clauses C_i must be kept in memory while deriving the other clause. This seems to be very different from how general resolution works with respect to space.

We now try to present our own intuition for what the correct lower bound on the refutation clause space of pebbling contradictions *should be*. Although the reasoning is quite informal and non-rigorous, our hope is that it will help the reader to navigate the formal proofs that will follow.

As we noted above, the resolution refutation of $Peb^2_{T_h}$ in [40] used to prove the $\frac{2}{3}h + \mathrm{O}(1)$ upper bound for binary tree pebbling contradictions is structurally quite similar to the optimal black-white pebbling of T_h presented in [55], and it somehow feels implausible that any resolution refutation would be able to do significantly better. This raises the suspicion that the black-white pebbling price $\mathsf{\textit{BW-Peb}}(G)$ might be a lower bound for $Sp(Peb^d_G \vdash 0)$ as long as $d \geq 2$. This suspicion is somewhat strengthened by the fact that for variable space, we do have such a lower bound in terms of black-white pebbling price.[2]

Theorem 5.11 ([16]). *For any $d \in \mathbb{N}^+$, $VarSp(Peb^d_G \vdash 0) \geq \mathsf{\textit{BW-Peb}}(G)$.*

If the refutation clause space of pebbling contradictions for general DAGs would be constant, Theorem 5.11 would imply that as $\mathsf{\textit{BW-Peb}}(G)$ grows larger, the clauses in memory get wider, and thus weaker. Still it would somehow be possible to derive a contradiction from a very small number of these clauses of unbounded width. This appears counterintuitive.

On the other hand, for one variable per vertex, i.e., pebbling degree $d = 1$, refutations of Peb^1_G in constant space have exactly these "counterintuitive" properties. The resolution refutation of Peb^1_G in Theorem 5.8 is constructed by first downloading the pebbling axiom for the target z and then moving the false literals downwards by resolving with pebbling axioms for vertices $v \in V(G) \setminus S$ in reverse topological order. This finally yields a clause $\bigvee_{v \in S} \overline{x(v)}_1 \vee x(z)_1$ of width $|S| + 1$, which can be eliminated by resolving with the source axioms $x(v)_1$ one by one for all $v \in S$ and then with the target axiom $\overline{x(z)}_1$ to yield the empty clause 0.

If we want to establish a non-constant lower bound on $Sp(Peb^d_G \vdash 0)$ for $d \geq 2$, we have to pin down why this case is different. Intuitively, the difference is that with only one variable per vertex, a single clause $\overline{x(v_1)}_1 \vee \ldots \vee \overline{x(v_m)}_1$ can express the disjunction of the falsity of an arbitrary number of vertices $v_1, \ldots, v_m$, but for $d = 2$, the straightforward way of expressing that both variables $x(v_i)_1$ and $x(v_i)_2$ are false for at least one out of m vertices requires 2^m clauses.

As was argued in Chapter 3, to prove a lower bound on the refutation clause space of pebbling contradictions it seems natural to try to interpret resolution refutations of Peb^d_G in terms of pebblings of the underlying graph G. Let us say that a vertex v is "true" if $\bigvee_{i=1}^{d} x(v)_i$ has been derived and "false" if $\overline{x(v)}_i$ has been

[2] To be precise, the result in [16] is stated for $d = 1$, but the proof generalizes easily to any $d \in \mathbb{N}^+$.

derived for all $i \in [d]$. Any resolution proof refutes a pebbling contradiction by deriving that some vertex v is both true and false and then resolving to get 0. Let w be any vertex with predecessors u, v. Then we can see that if we have derived that u and v are true, by downloading $\overline{x(u)_i} \lor \overline{x(v)_j} \lor \bigvee_{l=1}^{d} x(w)_l$ for all $i, j \in [d]$ we can derive $\bigvee_{l=1}^{d} x(w)_l$. This appears analogous to the rule that if u and v are black-pebbled we can place a black pebble on w. In the opposite direction, if we know $\overline{x(w)_l}$ for all $l \in [d]$, using the axioms $\overline{x(u)_i} \lor \overline{x(v)_j} \lor \bigvee_{l=1}^{d} x(w)_l$ we can derive that either u or v is false. This looks similar to eliminating a white pebble on w by placing white pebbles on the predecessors u and v, and then removing the pebble from w. Generalizing this loose, intuitive reasoning, we argue that a set of black-pebbled vertices V should correspond to the derived conjunction of truth of all $v \in V$, and that a set of white-pebbled vertices W should correspond to the derived disjunction of falsity of some $w \in W$.

Suppose that we could show that as the resolution derivation proceeds, the black and white pebbles corresponding to different clause configurations as outlined above move about on the vertices of G in accordance with the rules of the pebble game. If so, we would get that there is some clause configuration $\mathbb{C}$ corresponding to a lot of pebbles. This could in turn hopefully yield a lower bound for the refutation clause space. For if $\mathbb{C}$ corresponds to N black pebbles, i.e., implies N disjoint clauses, it seems likely that $|\mathbb{C}|$ should be linear in N. And if $\mathbb{C}$ corresponds to N white pebbles, $|\mathbb{C}|$ should grow with N if $d \geq 2$, since $\mathbb{C}$ has to force d literals false simultaneously for one out of N vertices.

This is the guiding intuition that served as a starting point for proving the results in this thesis. And although quite a few complications arise along the way, we believe that it is important when reading the thesis not to let all technical details obscure the rather simple intuitive correspondence sketched above.

Chapter 6

Pebbling Price of Layered Graphs

A key component in the proofs in Chapters 8 and 9 are the upper and lower bounds on pebbling price for binary trees and pyramid graphs. In this chapter, we prove the bounds that we will need on pebbling price for a more general class of so-called layered graphs, that includes trees and pyramids. These results are not new, but we present somewhat simplified proofs that might be of some independent interest.

6.1 Some Graph Notation and Terminology

Let us first present some notation and terminology that will be used in what follows. See Figure 6.1 for an illustration of the next definition.

Definition 6.1. We let $\mathit{succ}(v)$ denote the immediate successors and $\mathit{pred}(v)$ denote the immediate predecessors of a vertex v in a DAG G. Taking the transitive closures of $\mathit{succ}(\cdot)$ and $\mathit{pred}(\cdot)$, we let G^{∇}_{v} denote all vertices reachable from v (vertices "above" v) and G^{v}_{Δ} denote all vertices from which v is reachable (vertices "below" v). We write $G^{\not v}_{\Delta}$ and $G^{\nabla}_{\not v}$ to denote the corresponding sets with the vertex v itself removed. If $u, w \in \mathit{pred}(v)$, we say that u and w are *siblings*. If $u \notin G^{w}_{\Delta}$ and $w \notin G^{u}_{\Delta}$, we say that u and w are *non-comparable* vertices. Otherwise they are *comparable*.

When reasoning about arbitrary vertices we will often use as a canonical example a vertex r with assumed predecessors $\mathit{pred}(r) = \{p, q\}$.

Note that for a leaf v we have $\mathit{pred}(v) = \emptyset$, and for the sink z of G we have $\mathit{succ}(z) = \emptyset$. Also note that G^{v}_{Δ} and G^{∇}_{v} are sets of vertices, not subgraphs. However, we will allow ourselves to overload the notation and sometimes use this notation both for the subgraph and its vertices. Moreover, as a rule we will overload the notation for the graph G itself and its vertices, and usually write only G when we mean $V(G)$, and when this should be clear from context.

Definition 6.2 (Layered DAG). A *layered DAG* G is a DAG whose vertices are partitioned into (nonempty) sets of *layers* $V_0, V_1, \ldots, V_h$ on *levels* $0, 1, \ldots, h$, and

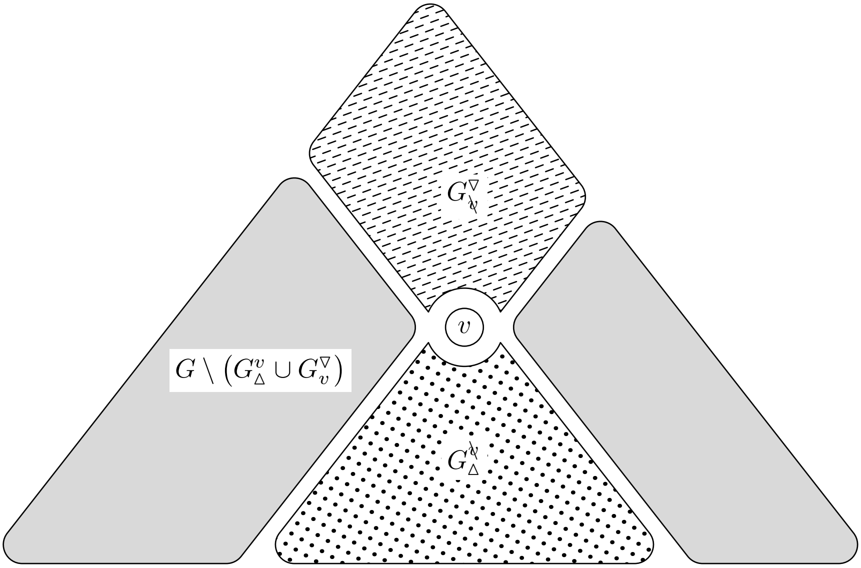

$G^{\triangledown}_{v}$
v
$G \setminus \left(G^{v}_{\triangle} \cup G^{\triangledown}_{v}\right)$
$G^{v}_{\triangle}$

Figure 6.1: Notation for sets of vertices in DAG G with respect to vertex v.

whose edges run between consecutive layers. That is, if (u, v) is a directed edge, then the level of u is $L - 1$ and the level of v is L for some $L \in [h]$. We say that h is the *height* of the layered DAG G.

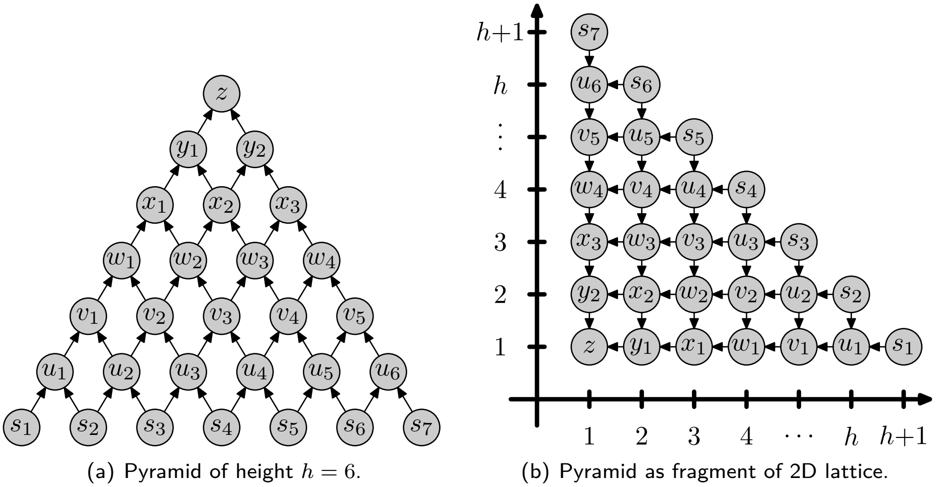
For the layered DAGs G that we will study, we will assume that all sources are on level 0, that all non-sources have indegree 2, and that there is a a unique sink z. The following notation will be convenient.

Definition 6.3 (Layered DAG notation). For a vertex u in a layered DAG G we let $\mathrm{level}(u)$ denote the level of u. For a vertex set U we let $\mathrm{minlevel}(U) = \min\{\mathrm{level}(u) : u \in U\}$ and $\mathrm{maxlevel}(U) = \max\{\mathrm{level}(u) : u \in U\}$ denote the lowest and highest level, respectively, of any vertex in U. Vertices in U on particular levels are denoted as follows:

- $U\{\succeq j\} = \{u \in U \mid \mathrm{level}(u) \geq j\}$ denotes the subset of all vertices in U on level j or higher.
- $U\{\succ j\} = \{u \in U \mid \mathrm{level}(u) > j\}$ denotes the vertices in U strictly above level j.
- $U\{\sim j\} = U\{\succeq j\} \setminus U\{\succ j\}$ denotes the vertices exactly on level j.

The vertex sets $U\{\preceq j\}$ and $U\{\prec j\}$ are defined completely analogously.

Let us next give the formal definition of a particular class of layered DAGs that we will be interested in, namely pyramids.



(a) Pyramid of height $h = 6$.

(b) Pyramid as fragment of 2D lattice.

Figure 6.2: Running example pyramid Π_6 of height 6 with labelled vertices.

Definition 6.4 (Pyramid graph). The *pyramid graph* Π_h of height h is a layered DAG with $h+1$ levels, where there is one vertex on the highest level (the sink z), two vertices on the next level et cetera down to $h+1$ vertices at the lowest level 0. The ith vertex at level L has incoming edges from the ith and $(i+1)$st vertices at level $L-1$.

Although most of what will be said in what follows holds for arbitrary layered DAGs, in this chapter we will tend to focus on pyramids. Figure 6.2(a) presents a pyramid graph with labelled vertices that we will use as a running example. Pyramid graphs can also be visualized as triangular fragments of a directed two-dimensional rectilinear lattice. In Figure 6.2(b), the pyramid in Figure 6.2(a) is redrawn as such a lattice fragment.

We also need some notation for contiguous and non-contiguous topologically ordered sets of vertices in a DAG.

Definition 6.5 (Paths and chains). We say that V is a *(totally) ordered* set of vertices in a DAG G, or a *chain*, if all vertices in V are comparable (i.e., if for all $u, v \in V$, either $u \in G^v_\vartriangle$ or $v \in G^u_\vartriangle$ holds). A *path* P is a contiguous chain, i.e., such that *succ*$(v) \cap P \neq \emptyset$ for all $v \in P$ except the top vertex.

We write $P : v \rightsquigarrow w$ to denote a path starting in v and ending in w. A *source path* is a path that starts at some source vertex of G. A *path via* w is a path such that $w \in P$. We will also say that P *visits* w. For a chain V, we let

- $\mathrm{bot}(V)$ denote the bottom vertex of V, i.e., the unique $v \in V$ such that $V \subseteq G^\triangledown_v$,
- $\mathrm{top}(V)$ denote the top vertex of V, i.e., the unique $v \in V$ such that $V \subseteq G^v_\vartriangle$,

- $\mathfrak{P}_{\text{in}}(V)$ denote the set of all paths $P : \text{bot}(V) \rightsquigarrow \text{top}(V)$ *via* V or *agreeing with* V, i.e., such that $V \subseteq P$, and
- $\mathfrak{P}_{\text{via}}(V)$ denote the set of all source paths *agreeing with* V.

We write $\bigcup \mathfrak{P}_{\text{in}}(V)$ to denote the union of the vertices in all paths $P \in \mathfrak{P}_{\text{in}}(V)$ and $\bigcup \mathfrak{P}_{\text{via}}(V)$ for the union of all vertices in paths $P \in \mathfrak{P}_{\text{via}}(V)$.

Unless otherwise stated, in the rest of this chapter G denotes a layered DAG; u, v, w, x, y denote vertices of G; U, V, W, X, Y denote sets of vertices; P denotes a path; and $\mathfrak{P}$ denotes a set of paths.

6.2 Pebbling Price of Binary Trees

Recall that T_h denotes the complete binary tree of height h considered as a DAG with edges directed towards the root. The fact that $\mathsf{Peb}(T_h) = h + 2$ can be established by induction over the tree height. We omit the easy proof. To prove the claim about black-white pebbling price in Theorem 5.2, it is convenient to generalize the definition of black-white pebbling somewhat.

Definition 6.6 (Visiting pebbling). Suppose that G is a DAG with sources S and sinks Z (one or many). Let the pebble game rules be as in Definition 5.1, and define cost of pebble configurations and pebblings in the same way.

A complete black-white pebbling *visiting* Z is a pebbling $\mathcal{P} = \{\mathbb{P}_0, \ldots, \mathbb{P}_\tau\}$ such that $\mathbb{P}_0 = \mathbb{P}_\tau = (\emptyset, \emptyset)$ and such that for every $z \in Z$, there exists a time $t_z \in [\tau]$ such that $z \in B_{t_z} \cup W_{t_z}$. The minimum cost of such a visiting pebbling is denoted $\mathsf{BW\text{-}Peb}^{\emptyset}(G)$.

It is easy to see that if $Z = \{z\}$ is a single sink, then it holds that $\mathsf{BW\text{-}Peb}^{\emptyset}(G) \leq \mathsf{BW\text{-}Peb}(G) \leq \mathsf{BW\text{-}Peb}^{\emptyset}(G) + 1$.

Using Definition 6.6, we can prove the following statement.

Theorem 6.7. *$\mathsf{BW\text{-}Peb}(T_h) = \lfloor \frac{h}{2} \rfloor + 3$ and $\mathsf{BW\text{-}Peb}^{\emptyset}(T_h) = \lfloor \frac{h-1}{2} \rfloor + 3$ for complete binary trees T_h of height $h \geq 1$.*

The proof of Theorem 6.7 is facilitated by the following proposition.

Proposition 6.8 ([34]). *Suppose that $\mathcal{P}$ is a black-white pebbling from (B_1, W_1) to (B_2, W_2). Then we can get a dual pebbling $\overline{\mathcal{P}}$ from (W_2, B_2) to (W_1, B_1) in exactly the same cost by reversing the sequence of moves and switching the colours of the pebbles.*

This proposition is an immediate consequence of the anti-symmetric nature of the pebbling rules in Definition 5.1 (just observe that the rules for placing and removing a black pebble are the duals of the rules for removing and placing a white pebble, respectively).

Proof of Theorem 6.7. Throughout this proof, we let z_1, z_2 denote the immediate predecessors of the root z of the tree.

We first show that $\mathsf{BW\text{-}Peb}^{\emptyset}(T_{h+2}) \geq \mathsf{BW\text{-}Peb}^{\emptyset}(T_h) + 1$. Suppose not, and let $\mathcal{P}$ be a pebbling in cost $K = \mathsf{BW\text{-}Peb}^{\emptyset}(T_h)$ for T_{h+2} making the minimum number of pebbling moves. Let T_h^i, $i \in [4]$, be the four disjoint subtrees of height h in T_{h+2}. It is easy to see that $\mathcal{P}$ restricted to $V(T_h^i)$ yields a legal pebbling of T_h^i visiting its root. It follows that there must exist distinct times t_i, $i \in [4]$, when T_h^i contains K pebbles and the rest of T_{h+2} is empty. Number the subtrees so that $t_1 < t_2 < t_3 < t_4$.

Suppose that the root z of T_{h+2} has been pebbled before time t_3. Then we can get a shorter pebbling of T_{h+2} by completing the subpebbling of T_h^3 but ignoring pebbling moves outside T_h^3 after time t_3.

Consequently, z must be pebbled for the first time after t_3. But at time t_3 the rest of the tree is empty, so in that case we can get a shorter legal pebbling by ignoring all moves outside T_h^3 before time t_3 and performing all moves in $\mathcal{P}$ after time t_3. Contradiction. Thus $\mathsf{BW\text{-}Peb}^{\emptyset}(T_{h+2}) \geq \mathsf{BW\text{-}Peb}^{\emptyset}(T_h) + 1$.

Next, it is easy to see that $\mathsf{BW\text{-}Peb}^{\emptyset}(T_{h+1}) \leq \mathsf{BW\text{-}Peb}(T_h)$. First black-pebble z_1 using a pebbling $\mathcal{P}$ in cost $\mathsf{BW\text{-}Peb}(T_h)$. Place white pebbles on z and z_2, and then remove the pebbles from z_1 and z. Finally, use the dual pebbling $\overline{\mathcal{P}}$ to get the white pebble off z_2 in the same cost $\mathsf{BW\text{-}Peb}(T_h)$.

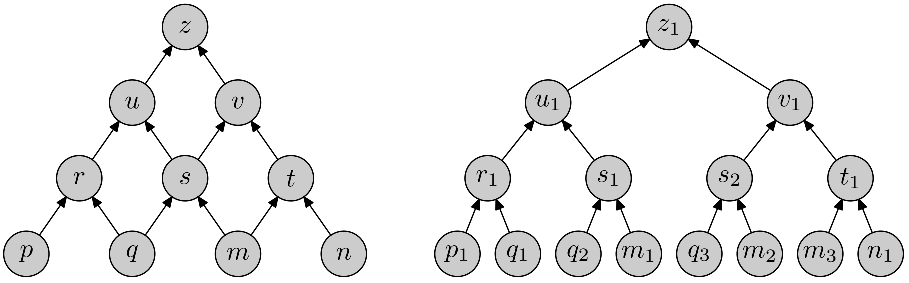
Since $\mathsf{BW\text{-}Peb}(T_1) = \mathsf{BW\text{-}Peb}^{\emptyset}(T_1) = 3$, it follows that $\mathsf{BW\text{-}Peb}^{\emptyset}(T_h) \geq \left\lfloor \frac{h-1}{2} \right\rfloor + 3$ and $\mathsf{BW\text{-}Peb}(T_h) \geq \left\lfloor \frac{(h+1)-1}{2} \right\rfloor + 3 = \left\lfloor \frac{h}{2} \right\rfloor + 3$. It remains to demonstrate that there are pebblings meeting these lower bounds. We construct such pebblings inductively.

Suppose for h odd that $\mathsf{BW\text{-}Peb}(T_h) = \mathsf{BW\text{-}Peb}^{\emptyset}(T_h) = \left\lfloor \frac{h-1}{2} \right\rfloor + 3 = \left\lfloor \frac{h}{2} \right\rfloor + 3$. Using the same pebbling as above for T_{h+1}, it is easy to see that $\mathsf{BW\text{-}Peb}^{\emptyset}(T_{h+1}) = \left\lfloor \frac{h}{2} \right\rfloor + 3$, and since the pebbling cost cannot increase by more than one when the height is increased by one we get $\mathsf{BW\text{-}Peb}^{\emptyset}(T_{h+2}) = \left\lfloor \frac{h}{2} \right\rfloor + 4 = \left\lfloor \frac{h+1}{2} \right\rfloor + 3$. In the same way we get $\mathsf{BW\text{-}Peb}(T_{h+1}) = \left\lfloor \frac{h+1}{2} \right\rfloor + 3$.

To pebble T_{h+2} in cost $\left\lfloor \frac{h+1}{2} \right\rfloor + 3$ leaving a pebble on z, first black-pebble the root z_1 of the subtree T_{h+1}^1 in cost $\left\lfloor \frac{h+1}{2} \right\rfloor + 3$. Leaving the pebble on z_1, make a pebbling visiting the root z_2 of T_{h+1}^2 in cost $\left\lfloor \frac{h}{2} \right\rfloor + 3 = \left\lfloor \frac{h+1}{2} \right\rfloor + 2$ using the pebbling for T_{h+1}^2 constructed inductively. In this pebbling there is a time t when z_2 is pebbled and T_{h+1}^2 contains at most $\left\lfloor \frac{h+1}{2} \right\rfloor + 1$ pebbles. At this time t, place a black pebble on z and remove the black pebble on z_1 without exceeding the total limit of $\left\lfloor \frac{h+1}{2} \right\rfloor + 3$ pebbles on T_{h+2}. Then finish the pebbling of T_{h+1}^2. The theorem follows. □

We can use our knowledge of the black and black-white pebbling price of binary trees to get upper bounds on pebbling price for any layered DAG.

Lemma 6.9. *For any layered DAG G_h of height h with a unique sink z and all non-sources having vertex indegree 2, it holds that $\mathsf{Peb}(G_h) \leq h + \mathrm{O}(1)$ and $\mathsf{BW\text{-}Peb}(G_h) \leq h/2 + \mathrm{O}(1)$.*



(a) Pyramid graph Π_3 of height 3.

(b) Binary tree T_3 with vertex labels from Π_3.

Figure 6.3: Binary tree with vertices labelled by pyramid as in proof of Lemma 6.9.

Proof. The bounds above are true for complete binary trees of height h, as we have just seen. It is not hard to see that the corresponding pebbling strategies can be used to pebble any layered graph of the same height with at most the same amount of pebbles.

Formally, suppose that the sink z of the DAG G_h has predecessors x and y. Label the root of T_h by z_1 and its predecessors by x_1 and y_1. Recursively, for a vertex in T_h labelled by w_i, look at the corresponding vertex w in G_h and suppose that $pred(w) = \{u, v\}$. Then label the vertices $pred(w_i)$ in T_h by u_j and v_k for the smallest positive indices j, k such that there are not already other vertices in T_h labelled u_j and v_k. In Figure 6.3 there is an illustration of how the vertices in a pyramid Π_3 of height 3 are mapped to vertices in the complete binary tree T_3 in this manner.

The result is a labelling of T_h where every vertex v in G_h corresponds to one or more distinct vertices $v_1, \ldots, v_{k_v}$ in T_h, and such that if $pred(w_i) = \{u_j, v_k\}$ in T_h, then $pred(w) = \{u, v\}$ in G_h. Given a pebbling strategy $\mathcal{P}$ for T_h, we can pebble G_h with at most the same amount of pebbles by mimicking any move on any v_i in T_h by performing the same move on v in G_h. The details are easily verified. □

6.3 The Black Pebbling Price of Pyramids

In this chapter, we will identify some layered graphs G_h for which the bound in Lemma 6.9 is also the asymptotically correct lower bound. As a warm-up, and also to introduce some important ideas, let us consider the black pebbling price of the pyramid Π_h of height h.

Theorem 6.10 ([32]). *Peb*$(\Pi_h) = h + 2$ *for* $h \geq 1$.

To prove this lower bound, it turns out that it is sufficient to study blocked paths in the pyramid.

Definition 6.11. A vertex set U *blocks* a path P if $U \cap P \neq \emptyset$. U blocks a set of paths $\mathfrak{P}$ if U blocks all $P \in \mathfrak{P}$.

Proof of Theorem 6.10. It is easy to devise (inductively) a black pebbling strategy that uses $h+2$ pebbles (using, for instance, Lemma 6.9). We show that this is also a lower bound.

Consider the first time t when all possible paths from sources to the sink are blocked by black pebbles. Suppose that P is (one of) the last path(s) blocked. Note that P must be blocked by a pebble placement on some source vertex u, since otherwise both vertices in $\mathit{pred}(u)$ would have to have pebbles on them and so P would already be blocked. The path P contains $h+1$ vertices, and for each vertex $v \in P \setminus \{u\}$ there is a unique path P_v that coincides with P from v onwards to the sink but arrives at v in a straight line from a source "in the opposite direction" of that of P, i.e., via the immediate predecessor of v not contained in P. At time $t-1$ all such paths $\{P_v \mid v \in P \setminus \{u\}\}$ must already be blocked, and since P is still open no pebble can block two paths $P_v \neq P_{v'}$ for $v, v' \in P \setminus \{u\}$, $v \neq v'$. Thus at time t there are at least $h+1$ pebbles on Π_h. Furthermore, without loss of generality each pebble placement on a source vertex is followed by another pebble placement (otherwise perform all removals immediately following after time t before making the pebble placement at time t). Thus at time $t+1$ there are $h+2$ pebbles on Π_h. □

We will repeatedly use the idea in the proof above about a set of paths converging at different levels to another fixed path, so we write it down as a separate observation.

Observation 6.12. *Suppose that u and w are vertices in Π_h on levels $L_u < L_w$ and that $P : u \rightsquigarrow w$ is a path from u to w. Let $K = L_w - L_v$ and write $P = \{v_0 = u, v_1, \ldots, v_K = w\}$. Then there is a set of K paths $\mathfrak{P} = \{P_1, \ldots, P_K\}$ such that P_i coincides with P from v_i onwards to w but arrives to v_i in a straight line from a source vertex via the immediate predecessor of v_i which is not contained in P, i.e., is distinct from v_{i-1}. In particular, for any i, j with $1 \leq i < j \leq k$ it holds that $P_i \cap P_j \subseteq P_j \cap P \subseteq P \setminus \{u\}$.*

We will refer to the paths $P_1, \ldots, P_K$ as a set of *converging source paths*, or just *converging paths*, for $P : u \rightsquigarrow w$. See Figure 6.4 for an example.

6.4 Black-White Pebbling Pyramids—a First Bound

For the black-white pebble game, Cook and Sethi proved the following lower bound on the pebbling price of pyramids.

Theorem 6.13 ([34]). *$\mathit{BW\text{-}Peb}(\Pi_h) \geq \frac{1}{2}\sqrt{h}$.*

In this section, we give a rather detailed exposition of the proof of this theorem, in the hope that this will provide helpful intuition for the tighter (and more intricate) lower bound proof that will follow in the next section.

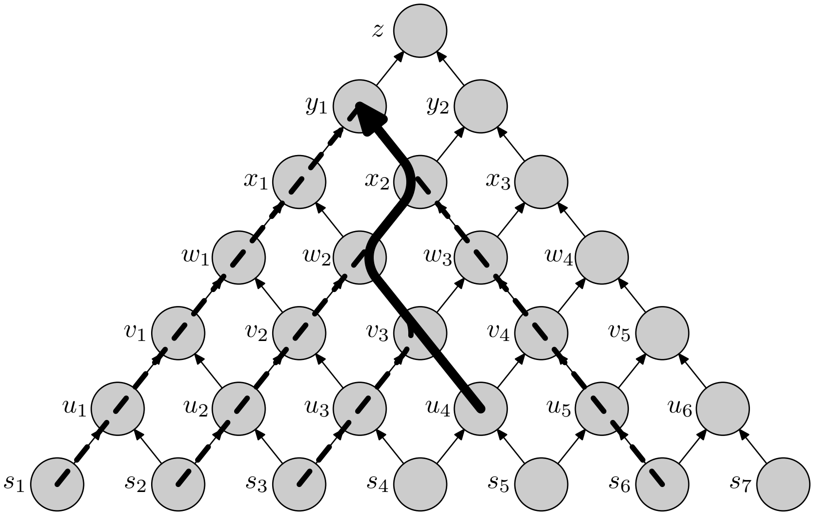


Figure 6.4: Set of converging source paths (dashed) for the path $P : u_4 \rightsquigarrow y_1$ (solid).

Cook and Sethi get their bound by proving something slightly stronger than in the statement of Theorem 6.13, namely the following.

Theorem 6.14 ([34]). *Suppose that $(B_0, \emptyset)$ and $(\{z\}, W_\tau)$ are pebble configurations in a pyramid Π_h such that there is a path $P : v \rightsquigarrow z$ from a source vertex v to the sink z with $P \cap (B_0 \cup W_\tau) = \emptyset$. Then for any conditional pebbling $\mathcal{P}$ from $(B_0, \emptyset)$ to $(\{z\}, W_\tau)$ it holds that $\mathsf{cost}(\mathcal{P}) \geq \frac{1}{2}\sqrt{h}$.*

That is, we start with a possibly non-empty set of black pebbles on B_0 and end with a possibly non-empty set of white pebbles on W_τ, but we have somehow managed to place a black pebble on the sink z and clear a path from a source to z that was not blocked when we started and is not blocked when we end. It is clear that Theorem 6.13 follows from this by taking $B_0 = W_\tau = \emptyset$.

In order to prove Theorem 6.14, we will use the anti-symmetry property of the black-white pebble game in Proposition 6.8 again. Note that, in particular, Proposition 6.8 implies that if B_0 and W_τ are such that there is a path $P : v \rightsquigarrow z$ from a source vertex v to the sink z with $P \cap (B_0 \cup W_\tau) = \emptyset$, then the cost of pebblings from $(B_0, \emptyset)$ to $(\{z\}, W_\tau)$, i.e., pebblings *placing a black pebble* on the sink of the pyramid, will be equal to the cost of pebblings from $(B_0, \{z\})$ to $(\emptyset, W_\tau)$, i.e., pebblings *removing a white pebble* from the sink of the pyramid.

We also need the technical observation that if we have a pebbling on a large graph and then look at a smaller subgraph and the moves in the pebbling restricted to vertices in this subgraph, we get a legal pebbling of this smaller graph. This is easy to verify directly from Definition 5.1.

Proposition 6.15. *Suppose that $\mathcal{P}$ is a pebbling on a DAG G and that G' is the induced subgraph on any connected set of vertices in $V(G)$. Then the pebbling $\mathcal{P}'$ where we only perform the moves in $\mathcal{P}$ concerning the vertices in G' is a legal pebbling.*

We are now ready to present the proof of Theorem 6.14.

Proof of Theorem 6.14. The proof is by induction over the height h of the pyramid Π. For the induction base, note that it is obvious for any pyramid of height $h \geq 1$ that $\mathit{cost}(\mathcal{P}) \geq 3$. The sink z has a black pebble at time τ. At the time when this pebble was placed on z, both its predecessors must also have had pebbles on them. Thus for $h \leq 36$ we have $\mathit{cost}(\mathcal{P}) \geq \frac{1}{2}\sqrt{h}$.

Suppose that the statement in the theorem is true for all $h' < h$. By the anti-symmetry in Proposition 6.8 we can assume that the bound holds for *all* pebblings $\mathcal{P}$ such that there is a path $P : v \rightsquigarrow z$ from a source vertex v to the sink z with $P \cap (B_0 \cup W_\tau) = \emptyset$, and such that $\mathcal{P}$ either leaves a black pebble on z or removes an initial white pebble from z.

Now we do the induction step. Suppose for a pyramid Π of height h that $\mathcal{P}$ is a pebbling in minimum cost from $(B_0, \emptyset)$ to $(\{z\}, W_\tau)$. Without loss of generality, we can assume that $\mathcal{P}$ is a pebbling with the least number of pebbling moves among such minimum-cost pebblings.

Consider the last time σ when we have a configuration (B_σ, W_σ) such that $B_\sigma \cup W_\tau$ blocks all paths from sources to z, but this is not true for $B_{\sigma-1} \cup W_\tau$. Clearly, there is such time σ since $B_0 \cup W_\tau$ does not block all paths to z but $B_\tau \cup W_\tau = \{z\} \cup W_\tau$ trivially does.

Since $B_{\sigma-1} \cup W_\tau$ does not block all paths from sources to z, the move at time σ must be a placement of a black pebble on some vertex r. Also, if r is not a source there must exist at least one white-pebbled predecessor q of r and a path P from a source via q and r to z such that P is not blocked by $B_{\sigma-1} \cup W_\tau$. (Both predecessors must have pebbles at time $\sigma - 1$, but if both predecessors were black-pebbled, $B_{\sigma-1} \cup W_\tau$ would already block all paths.)

We claim that if r is at distance $2\sqrt{h}$ or more from the sink, then we are done. For consider the converging paths of Observation 6.12 for the subpath $P^r : r \rightsquigarrow z$ of P. All these paths must be blocked by $(B_\sigma \cup W_\tau) \setminus \{r\} = B_{\sigma-1} \cup W_\tau$ but there are no pebbles from $B_{\sigma-1} \cup W_\tau$ on P^r, so $|B_{\sigma-1} \cup W_\tau| \geq 2\sqrt{h}$ yielding a pebbling cost of at least $\max\{|B_{\sigma-1}|, |W_\tau|\} \geq \sqrt{h}$. Suppose therefore that r is at distance strictly less than $2\sqrt{h}$ from the sink.

Consider the subpyramid Π^q_Δ rooted at q. The height of Π^q_Δ is at least $h - 2\sqrt{h}$. At time σ there is a white pebble at the sink q and a path P' from a source to q (namely the subpath $P' = P \cap \Pi^q_\Delta$ of P) such that $B_\sigma \cup W_\tau$ does not block P'. Looking just at the pebbles inside Π^q_Δ during the time interval $[\sigma, \tau]$ and using Proposition 6.15, we see that we get a pebbling removing the white pebble on q and opening a path to q in Π^q_Δ. By the induction hypothesis (and anti-symmetry), this pebbling inside Π^q_Δ costs at least $\frac{1}{2}\sqrt{h - 2\sqrt{h}} \geq \frac{1}{2}\sqrt{h} - 1$ if $h \geq 4$.

Also, we claim that during the whole time interval $[\sigma, \tau]$ there must be a pebble in Π outside Π^q_Δ. This is true at time σ and at time τ. Suppose there is some time $t \in (\sigma, \tau)$ such that all pebbles are inside Π^q_Δ. Then it is easy to verify that we get a correct pebbling $\mathcal{P}'$ of all of Π by ignoring all pebble placements and removals outside Π^q_Δ before time t. This pebbling $\mathcal{P}'$ has at most the same cost as $\mathcal{P}$ and has strictly fewer moves (since it does not place a black pebble on r at time σ, for instance), contradicting the assumed minimality of $\mathcal{P}$. Thus there is at least one pebble outside Π^q_Δ during the whole time interval $[\sigma, \tau]$, so the total cost of $\mathcal{P}$ is at least $1 + (\frac{1}{2}\sqrt{h} - 1) = \frac{1}{2}\sqrt{h}$. The theorem follows. □

Reading the proof of Theorem 6.13, it is hard to avoid the feeling that this bound cannot be optimal. And indeed, Klawe [49] later proved that $\mathsf{BW\text{-}Peb}(\Pi_h) \geq \frac{h}{2} + \mathrm{O}(1)$. This result is tight, in view of Lemma 6.9, and it can be noted that the additive constants in the upper and lower bounds are off by at most one. As we will see in the next section, Klawe's proof is technically quite intricate. We presented the easier and perhaps more intuitive proof of Cook and Sethi first in the hope that this will have helped to prepare the reader for what is to follow.

6.5 A Tight Bound for Black-White Pebbling Layered DAGs

In this section, we give a detailed exposition of the lower bound in [49], in the process simplifying the proof somewhat. Much of the notation and terminology has been changed from [49] to fit better with this thesis. Also, it should be noted that we restrict all definitions to layered graphs, in contrast to Klawe who deals with a somewhat more general class of graphs. We concentrate on layered graphs mainly to avoid unnecessary complications in the exposition, and since it can be proven that no graphs in [49] can give a better size/pebbling price trade-off than one gets for layered graphs.

Recall from Definition 6.5 that a *path via w* is a path P such that $w \in P$. We will also say that *P visits w*. The notation $\mathfrak{P}_{\mathrm{via}}(w)$ is used to denote all source paths visiting w. Note that a path $P \in \mathfrak{P}_{\mathrm{via}}(w)$ visiting w may continue after w, or may end in w.

Definition 6.16 (Hiding set). A vertex set U *hides* a vertex w if U blocks all source paths visiting w, i.e., if U blocks $\mathfrak{P}_{\mathrm{via}}(w)$. U hides W if U hides all $w \in W$. If so, we say that U is a *hiding set* for W. We write $\llbracket U \rrbracket$ to denote the set of all vertices hidden by U.

Our perspective is that we are standing at the sources of G and looking towards the sink. Then *U hides w* if we "cannot see" w from the sources since U completely hides w. When *U blocks* a path P is is possible that we can "see" the beginning of the path, but we cannot walk all of the path since it is blocked somewhere on the way. The reason why this terminological distinction is convenient will become clearer in Chapter 9.

Note that if U should hide w, then in particular it must block all paths ending in w. Therefore, when looking at minimal hiding sets we can assume without loss of generality that no vertex in U is on a level higher than w.

It is an easy exercise to show that the hiding relation is transitive, i.e., that if U hides V and V hides W, then U hides W.

Proposition 6.17. *If $V \subseteq [\![U]\!]$ and $W \subseteq [\![V]\!]$, then $W \subseteq [\![U]\!]$.*

One key concept in Klawe's paper is that of *potential*. The potential of $\mathbb{P} = (B, W)$ is intended to measure how "good" the configuration $\mathbb{P}$ is, or at least how hard it is to reach in a pebbling. Note that this is not captured by the cost of the current pebble configuration. For instance, the final configuration $\mathbb{P}_\tau = (\{z\}, \emptyset)$ is the best configuration conceivable, but only costs 1. At the other extreme, the configuration $\mathbb{P}$ in a pyramid with, say, all vertices on level L white-pebbled and all vertices on level $L+1$ black-pebbled is potentially very expensive (for low levels L), but does not seem very useful. Since this configuration on the one hand is quite expensive, but on the other hand is extremely easy to derive (just white-pebble all vertices on level L, and then black-pebble all vertices on level $L+1$), here the cost seems like a gross overestimation of the "goodness" of $\mathbb{P}$.

Klawe's potential measure remedies this. The potential of a pebble configuration (B, W) is defined as the minimum measure of any set U that together with W hides B. Recall that $U\{\succeq j\}$ denotes the subset of all vertices in U on level j or higher in a layered graph G.

Definition 6.18 (Measure). The *jth partial measure* of the vertex set U in G is

$$m_G^j(U) = \begin{cases} j + 2|U\{\succeq j\}| & \text{if } U\{\succeq j\} \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases}$$

and the *measure* of U is $m_G(U) = \max_j \{m_G^j(U)\}$.

Definition 6.19 (Potential). We say that U is a hiding set for a black-white pebble configuration $\mathbb{P} = (B, W)$ in a layered graph G if $U \cup W$ hides B. We define the *potential* of the pebble configuration to be

$$\mathrm{pot}_G(\mathbb{P}) = \mathrm{pot}_G(B, W) = \min\{m_G(U) : U \text{ is a hiding set for } (B, W)\} \ .$$

If U is a hiding set for (B, W) with minimal measure $m_G(U)$ among all vertex sets U' such that $U' \cup W$ hides B, we say that U is a *minimum-measure* hiding set for $\mathbb{P}$.

Remark 6.20. Klawe does not use the level of a vertex u in Definitions 6.18 and 6.19, but instead the black pebbling price $\mathsf{Peb}(\{u\}, \emptyset)$ of the configuration with a black pebble on u and no other pebbles in the DAG. For pyramids, these two concepts are equivalent, and we feel that the exposition can be made considerably simpler by using levels.

Since the graph under consideration will almost always be clear from context, we will tend to omit the subindex G in measures and potentials.

We remark that although this might not be immediately obvious, there is quite a lot of nice intuition why Definition 6.19 is a relevant estimation of how "good" a pebble configuration is. We refer the reader to Section 2 of [49] for a discussion about this. Let us just note that with this definition, the pebble configuration $\mathbb{P}_\tau = (\{z\}, \emptyset)$ has high potential, as we shall soon see, while the configuration with all vertices on level L white-pebbled and all vertices on level $L+1$ black-pebbled has potential zero.

Klawe proves two facts about the potentials of the pebble configurations in any black-white pebbling $\mathcal{P} = \{\mathbb{P}_0, \ldots, \mathbb{P}_\tau\}$ of a pyramid graph Π_h:

1. The potential correctly estimates the goodness of the current configuration $\mathbb{P}_t$ by taking into account the whole pebbling that has led to $\mathbb{P}_t$. Namely, $\mathrm{pot}(\mathbb{P}_t) \leq 2 \cdot \max_{s \leq t}\{\mathit{cost}(\mathbb{P}_s)\}$.

2. The final configuration $\mathbb{P}_\tau = (\{z\}, \emptyset)$ has high potential, namely $\mathrm{pot}(\{z\}, \emptyset) = h + \mathrm{O}(1)$.

Combining these two parts, one clearly gets a lower bound on pebbling price.

For pyramids, part 2 is not too hard to show directly. In fact, it is a useful exercise if one wants to get some feeling for how the potential works. Part 1 is much trickier. It is proven by induction over the pebbling. As it turns out, the whole induction proof hinges on the following key property.

Property 6.21 (Limited hiding-cardinality property). We say that the black-white pebble configuration $\mathbb{P} = (B, W)$ in G has the *Limited hiding-cardinality property*, or just the *LHC property* for short, if there is a vertex set U such that

1. U is a hiding set for $\mathbb{P}$,

2. $\mathrm{pot}_G(\mathbb{P}) = m(U)$,

3. $U = B$ or $|U| < |B| + |W| = \mathit{cost}(\mathbb{P})$.

We say that the graph G has the Limited hiding-cardinality property if all black-white pebble configurations $\mathbb{P} = (B, W)$ on G have the Limited hiding-cardinality property.

Note that requirements 1 and 2 just say that U is a vertex set that witnesses the potential of $\mathbb{P}$. The important point here is requirement 3, which says, basically, that if we are given a hiding set U with minimum measure but with size exceeding the cost of the black-white pebble configuration $\mathbb{P}$, then we can pick *another* hiding set U' which keeps the minimum measure but decreases the cardinality to at most $\mathit{cost}(\mathbb{P})$.

Given Property 6.21, the induction proof for part 1 follows quite easily. The main part of the paper [49] is then spent on proving that a class of DAGs including pyramids have Property 6.21. Let us see what the lower bound proof looks like, assuming that Property 6.21 holds.

Lemma 6.22 (Theorem 2.2 in [49]). *Let G be a layered graph possessing the LHC property and suppose that $\mathcal{P} = \{\mathbb{P}_0 = \emptyset, \mathbb{P}_1, \ldots, \mathbb{P}_\tau\}$ is any unconditional black-white pebbling on G. Then it holds for all $t = 1, \ldots, \tau$ that* $\mathrm{pot}_G(\mathbb{P}_t) \leq 2 \cdot \max_{s \leq t}\{\mathit{cost}(\mathbb{P}_s)\}$.

Proof. To simplify the proof, let us assume without loss of generality that no white pebble is ever removed from a source. If $\mathcal{P}$ contains such moves, we just substitute for each such white pebble placement on v a black pebble placement on v instead, and when the white pebble is removed we remove the corresponding black pebble. It is easy to check that this results in a legal pebbling $\mathcal{P}'$ that has exactly the same cost.

The proof is by induction. The base case $\mathbb{P}_0 = \emptyset$ is trivial. For the induction hypothesis, suppose that $\mathrm{pot}(\mathbb{P}_t) \leq 2 \cdot \max_{s \leq t}\{\mathit{cost}(\mathbb{P}_s)\}$ and let U_t be a vertex set as in Property 6.21, i.e., such that $U_t \cup W_t$ hides B_t, $\mathrm{pot}(\mathbb{P}_t) = m(U_t)$ and $|U_t| \leq \mathit{cost}(\mathbb{P}_t) = |B| + |W|$.

Consider $\mathbb{P}_{t+1}$. We need to show that $\mathrm{pot}(\mathbb{P}_{t+1}) \leq 2 \cdot \max_{s \leq t+1}\{\mathit{cost}(\mathbb{P}_s)\}$. By the induction hypothesis, it is sufficient to show that

$$\mathrm{pot}(\mathbb{P}_{t+1}) \leq \max\{\mathrm{pot}(\mathbb{P}_t), 2 \cdot \mathit{cost}(\mathbb{P}_{t+1})\} \ . \tag{6.1}$$

We also note that if $U_t \cup W_{t+1}$ hides B_{t+1} we are done, since if so $\mathrm{pot}(\mathbb{P}_{t+1}) \leq m(U_t) = \mathrm{pot}(\mathbb{P}_t)$. We make a case analysis depending on the type of move made to get from $\mathbb{P}_t$ to $\mathbb{P}_{t+1}$.

1. Removal of black pebble: In this case, $U_t \cup W_{t+1} = U_t \cup W_t$ obviously hides $B_{t+1} \subset B_t$ as well, so $\mathrm{pot}(\mathbb{P}_{t+1}) \leq \mathrm{pot}(\mathbb{P}_t)$.

2. Placement of white pebble: Again, $U_t \cup W_{t+1} \supset U_t \cup W_t$ hides $B_{t+1} = B_t$, so $\mathrm{pot}(\mathbb{P}_{t+1}) \leq \mathrm{pot}(\mathbb{P}_t)$.

3. Removal of white pebble: Suppose that a white pebble is removed from the vertex w, so $W_{t+1} = W_t \setminus \{w\}$. As noted above, without loss of generality w is not a source vertex. We claim that $U_t \cup W_{t+1}$ still hides $B_{t+1} = B_t$, from which $\mathrm{pot}(\mathbb{P}_{t+1}) \leq \mathrm{pot}(\mathbb{P}_t)$ follows as above.

   To see that the claim is true, note that $\mathit{pred}(w) \subseteq B_t \cup W_t$ by the pebbling rules, for otherwise we would not be able to remove the white pebble on w. If $\mathit{pred}(w) \subseteq W_t$ we are done, since then $U_t \cup W_{t+1}$ hides $U_t \cup W_t$ and we can use the transitivity in Proposition 6.17. If instead there is some $v \in \mathit{pred}(w) \cap B_t$, then $U_t \cup W_t = U_t \cup W_{t+1} \cup \{w\}$ hides v by assumption. Since w is a successor of v, and therefore on a higher level than v, we must have $U_t \cup W_t \setminus \{w\}$ hiding v. Thus in any case $U_t \cup W_{t+1}$ hides $\mathit{pred}(w)$, so by transitivity $U_t \cup W_{t+1}$ hides B_{t+1}.

4. Placement of black pebble: Suppose that a black pebble is placed on v. If v is not a source, by the pebbling rules we again have that $\mathit{pred}(v) \subseteq B_t \cup W_t$.

In particular, $B_t \cup W_t$ hides v and by transitivity we have that $U_t \cup W_{t+1} = U_t \cup W_t$ hides $B_t \cup \{v\} = B_{t+1}$.

The case when v is a source turns out to be the only interesting one. Now $U_t \cup W_t$ does not necessarily hide $B_t \cup \{v\} = B_{t+1}$ any longer. An obvious fix is to try with $U_t \cup \{v\} \cup W_t$ instead. This set clearly hides B_{t+1}, but it can be the case that $m(U_t \cup \{v\}) > m(U_t)$. This is problematic, since we could have $\mathrm{pot}(\mathbb{P}_{t+1}) = m(U_t \cup \{v\}) > m(U_t) = \mathrm{pot}(\mathbb{P}_t)$. And we do not know that the inequality $\mathrm{pot}(\mathbb{P}_t) \leq 2 \cdot \mathit{cost}(\mathbb{P}_t)$ holds, only that $\mathrm{pot}(\mathbb{P}_t) \leq 2 \cdot \max_{s \leq t}\{\mathit{cost}(\mathbb{P}_s)\}$. This means that it can happen that $\mathrm{pot}(\mathbb{P}_{t+1}) > 2 \cdot \mathit{cost}(\mathbb{P}_{t+1})$, in which case the induction step fails. However, we claim that using the Limited hiding-cardinality property 6.21 we can prove for $U_{t+1} = U_t \cup \{v\}$ that

$$m(U_{t+1}) = m(U_t \cup \{v\}) \leq \max\{m(U_t), 2 \cdot \mathit{cost}(\mathbb{P}_{t+1})\} \ , \tag{6.2}$$

which shows that (6.1) holds and the induction steps goes through.

Namely, suppose that U_t is chosen as in Property 6.21 and consider $U_{t+1} = U_t \cup \{v\}$. Then U_{t+1} is a hiding set for $\mathbb{P}_{t+1} = (B_t \cup \{v\}, W_t)$ and hence $\mathrm{pot}(\mathbb{P}_{t+1}) \leq m(U_{t+1})$. For $j > 0$, it holds that $U_{t+1}\{\succeq j\} = U_t\{\succeq j\}$ and thus $m^j(U_{t+1}) = m^j(U_t)$. On the bottom level, using that the inequality $|U_t| \leq \mathit{cost}(\mathbb{P}_t)$ holds by the LHC property, we have

$$m^0(U_{t+1}) = 2 \cdot |U_{t+1}| = 2 \cdot (|U_t| + 1) \leq 2 \cdot (\mathit{cost}(\mathbb{P}_t) + 1) = 2 \cdot \mathit{cost}(\mathbb{P}_{t+1}) \tag{6.3}$$

and we get that

$$\begin{aligned} m(U_{t+1}) &= \max_j\{m^j(U_{t+1})\} = \max\{\max_{j>0}\{m^j(U_t)\}, m^0(U_{t+1})\} \\ &\leq \max\{m(U_t), 2 \cdot \mathit{cost}(\mathbb{P}_{t+1})\} = \max\{\mathrm{pot}(\mathbb{P}_t), 2 \cdot \mathit{cost}(\mathbb{P}_{t+1})\} \end{aligned} \tag{6.4}$$

which is exactly what we need.

We see that the inequality (6.1) holds in all cases in our case analysis, which proves the lemma. □

The lower bound on black-white pebbling price now follows by showing that the final pebble configuration $(\{z\}, \emptyset)$ has high potential.

Lemma 6.23. *For z the sink of a pyramid Π_h of height h, the pebble configuration $(\{z\}, \emptyset)$ has potential $\mathrm{pot}_{\Pi_h}(\{z\}, \emptyset) = h + 2$.*

Proof. This follows easily from the Limited hiding-cardinality property (which says that U can be chosen so that either $U \subseteq \{z\}$ or $|U| \leq 0$), but let us show that this assumption is not necessary here. The set $U = \{z\}$ hides itself and has measure $m(U) = m^h(U) = h + 2 \cdot 1 = h + 2$. Suppose that z is hidden by some $U' \neq \{z\}$. Without loss of generality U' is minimal, i.e., no strict subset of U' hides z. Let u be a vertex in U' on minimal level $\mathrm{minlevel}(U) = L < h$. The fact that U' is minimal implies that there is a path $P : u \rightsquigarrow z$ such that $(P \setminus \{u\}) \cap U' = \emptyset$ (otherwise $U' \setminus \{u\}$ would hide z). By Observation 6.12, there must exist $h - L$ converging paths from sources to z that are all blocked by distinct pebbles in $U' \setminus \{u\}$. It follows that

$$m(U') \geq m^L(U') = L + 2\left|U'\{\succeq L\}\right| = L + 2\left|U'\right| \geq L + 2 \cdot (h+1-L) > h+2 \quad (6.5)$$

(where we used that $U'\{\succeq L\} = U'$ since $L = \mathrm{minlevel}(U)$). Thus $U = \{z\}$ is the unique minimum-measure hiding set for $(\{z\}, \emptyset)$, and the potential is $\mathrm{pot}(\{z\}, \emptyset) = h + 2$. □

Since [49] proves that pyramids possess the Limited hiding-cardinality property, and since there are pebblings that yield matching upper bounds, we have the following theorem.

Theorem 6.24 ([49]). $\mathit{BW\text{-}Peb}(\Pi_h) = \frac{h}{2} + \mathrm{O}(1)$.

Proof. The upper bound on the pebbling price was shown in Lemma 6.9. For the lower bound, Lemma 6.23 says that the final pebble configuration $(\{z\}, \emptyset)$ in any complete pebbling $\mathcal{P}$ of Π_h has potential $\mathrm{pot}(\{z\}, \emptyset) = h + 2$. According to Lemma 6.22, $\mathrm{pot}(\{z\}, \emptyset) \leq 2 \cdot \mathit{cost}(\mathcal{P})$. Thus $\mathit{BW\text{-}Peb}(\Pi_h) \geq h/2 + 1$. □

In the final two subsections of this section, we provide a fairly detailed overview of the proof that pyramids do indeed possess the Limited hiding-cardinality property. As was discussed above, the reason for giving all the details is that we will need to use and modify the construction in non-trivial ways in Chapter 9, where we will use ideas inspired by Klawe's paper to prove lower bounds on the pebbling price of pyramids in another pebble game.

6.5.1 Proving the Limited Hiding-Cardinality Property

We present the proof of that pyramids have the Limited hiding-cardinality property in a top-down fashion as follows.

1. First, we study what hiding sets look like in order to better understand their structure. Along the way, we make a few definitions and prove some lemmas culminating in Definition 6.30 and Lemma 6.34.

2. We conclude that it seems like a good idea to try to split our hiding set into disjoint components, prove the LHC property locally, and then add everything together to get a proof that works globally. We make an attempt to do this in Theorem 6.35, but note that the argument does not quite work. However, if we assume a slightly stronger property locally for our disjoint components (Property 6.37), the proof goes through.

3. We then prove this stronger local property by assuming that pyramid graphs have a certain *spreading* property (Definition 6.44 and Theorem 6.45), and by showing in Lemmas 6.43 and 6.46 that the stronger local property holds for such spreading graphs.

4. Finally, in Section 6.5.2, we give a simplified proof of the theorem in [49] that pyramids are indeed spreading.

From this, the desired conclusion follows.

For a start, we need two definitions. The intuition for the first one is that the vertex set U is *tight* if is does not contain any "unnecessary" vertex u hidden by the other vertices in U.

Definition 6.25 (Tight vertex set). The vertex set U is *tight* if for all $u \in U$ it holds that $u \notin \lceil\!\lceil U \setminus \{u\} \rceil\!\rceil$.

If x is a vertex hidden by U, we can identify a subset of U that is necessary for hiding x.

Definition 6.26 (Necessary hiding subset). If $x \in \lceil\!\lceil U \rceil\!\rceil$, we define $U\lfloor\!\lfloor x \rfloor\!\rfloor$ to be the subset of U such that for each $u \in U\lfloor\!\lfloor x \rfloor\!\rfloor$ there is a source path P ending in x for which $P \cap U = \{u\}$.

We observe that if U is tight and $u \in U$, then $U\lfloor\!\lfloor u \rfloor\!\rfloor = \{u\}$. This is not the case for non-tight sets. If we let $U = \{u\} \cup \mathit{pred}(u)$ for some non-source u, Definition 6.26 yields that $U\lfloor\!\lfloor u \rfloor\!\rfloor = \emptyset$. The vertices in $U\lfloor\!\lfloor x \rfloor\!\rfloor$ must be contained in every subset of U that hides x, since for each $v \in U\lfloor\!\lfloor x \rfloor\!\rfloor$ there is a source path to x that intersects U only in v. But if U is tight, the set $U\lfloor\!\lfloor x \rfloor\!\rfloor$ is also *sufficient* to hide x, i.e., $x \in \lceil\!\lceil U\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil$.

Lemma 6.27 (Lemma 3.1 in [49]). *If U is tight and $x \in \lceil\!\lceil U \rceil\!\rceil$, then $U\lfloor\!\lfloor x \rfloor\!\rfloor$ hides x and this set is also contained in every subset of U that hides x.*

Proof. The necessity was argued above, so the interesting part is that $x \in \lceil\!\lceil U\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil$. Suppose not. Let P_1 be a source path to x such that $P_1 \cap U\lfloor\!\lfloor x \rfloor\!\rfloor = \emptyset$. Since U hides x, U blocks P_1. Let v be the highest-level element in $P_1 \cap U$ (i.e., , the vertex on this path closest to x). Since U is tight, $U \setminus \{v\}$ does not hide v. Let P_2 be a source path to v such that $P_2 \cap (U \setminus \{v\}) = \emptyset$. Then going first along P_2 and switching to P_1 in v we get a path to x that intersects U only in v. But if so, we have $v \in U\lfloor\!\lfloor x \rfloor\!\rfloor$ contrary to assumption. Thus, $x \in \lceil\!\lceil U\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil$ must hold. □

Given a vertex set U, the tight subset of U hiding the same elements is uniquely determined.

Lemma 6.28. *For any vertex set U in a layered graph G there is a uniquely determined minimal subset $U^* \subseteq U$ such that $\lceil\!\lceil U^* \rceil\!\rceil = \lceil\!\lceil U \rceil\!\rceil$, U^* is tight, and for any $U' \subseteq U$ with $\lceil\!\lceil U' \rceil\!\rceil = \lceil\!\lceil U \rceil\!\rceil$ it holds that $U^* \subseteq U'$.*

Proof. We construct the set U^* bottom-up, layer by layer. We will let U_i^* be the set of vertices on level i or lower in the tight hiding set under construction, and U_i^r be the set of vertices in U strictly above level i remaining to be hidden.

Let $L = \mathrm{minlevel}(U)$. For $i < L$, we define $U_i^* = \emptyset$. Clearly, all vertices on level L in U must be present also in U^*, since no vertices in $U\{\succ L\}$ can hide these vertices and vertices on the same level cannot help hiding each other. Set $U_L^* = U\{\sim L\} = U \setminus U\{\succ L\}$. Now we can remove from U all vertices hidden by U_L^*, so set $U_L^r = U \setminus \lceil\!\lceil U_L^* \rceil\!\rceil$. Note that there are no vertices on or below level L left in U_L^r, i.e., $U_L^r = U_L^r\{\succ L\}$, and that U_L^* hides the same vertices as does $U\{\preceq L\}$ (since the two sets are equal).

Inductively, suppose we have constructed the vertex sets U_{i-1}^* and U_{i-1}^r. Just as above, set $U_i^* = U_{i-1}^* \cup U_{i-1}^r\{\sim i\}$ and $U_i^r = U_{i-1}^r \setminus \lceil\!\lceil U_i^* \rceil\!\rceil$. If there are no vertices remaining on level i to be hidden, i.e., if $U_{i-1}^r\{\sim i\} = \emptyset$, nothing happens and we get $U_i^* = U_{i-1}^*$ and $U_i^r = U_{i-1}^r$. Otherwise the vertices on level i in U_{i-1}^r are added to U_i^* and all of these vertices, as well as any vertices above in U_{i-1}^r now being hidden, are removed from U_{i-1}^r resulting in a smaller set U_i^r.

To conclude, we set $U^* = U_M^*$ for $M = \mathrm{maxlevel}(U)$. By construction, the invariant

$$\lceil\!\lceil U_i^* \rceil\!\rceil = \lceil\!\lceil U\{\preceq i\} \rceil\!\rceil \tag{6.6}$$

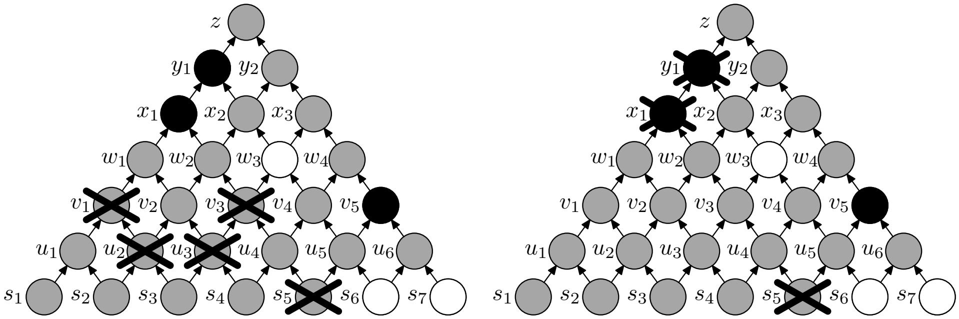
holds for all levels i. Thus, $\lceil\!\lceil U^* \rceil\!\rceil = \lceil\!\lceil U \rceil\!\rceil$. Also, U^* must be tight since if $v \in U^*$ and $\mathrm{level}(v) = i$, by construction $U^*\{\prec i\}$ does not hide v, and (as was argued above) neither does $U^*\{\succeq i\} \setminus \{v\}$. Finally, suppose that $U' \subseteq U$ is a hiding set for U with $U^* \nsubseteq U'$. Consider $v \in U^* \setminus U'$ and suppose $\mathrm{level}(v) = i$. On the one hand, we have $v \notin \lceil\!\lceil U_{i-1}^* \rceil\!\rceil$ by construction. On the other hand, by assumption it holds that $v \in \lceil\!\lceil U'\{\prec i\} \rceil\!\rceil$ and thus $v \in \lceil\!\lceil U\{\prec i\} \rceil\!\rceil$. But then by the invariant (6.6) we know that $v \in \lceil\!\lceil U_{i-1}^* \rceil\!\rceil$, which yields a contradiction. Hence, $U^* \subseteq U'$ and the lemma follows. □

We remark that U^* can in fact be seen to contain exactly those elements $u \in U$ such that u is not hidden by $U \setminus \{u\}$.

It follows from Lemma 6.28 that if U is a minimum-measure hiding set for $\mathbb{P} = (B, W)$, we can assume without loss of generality that $U \cup W$ is tight. More formally, if $U \cup W$ is not tight, we can consider minimal subsets $U' \subseteq U$ and $W' \subseteq W$ such that $U' \cup W'$ hides B and is tight, and prove the LHC property for B and W' with respect to this U' instead. Then clearly the LHC property holds also for B and W.

Suppose that we have a set U that together with W hides B. Suppose furthermore that B contains vertices very far apart in the graph. Then it might very well be the case that $U \cup W$ can be split into a number of disjoint subsets $U_i \cup W_i$ responsible for hiding different parts B_i of B, but which are wholly independent of one another. Let us give an example of this.

Example 6.29. Suppose we have the configuration of black and white pebbles $(B, W) = (\{x_1, y_1, v_5\}, \{w_3, s_6, s_7\})$ and the hiding set $U = \{v_1, u_2, u_3, v_3, s_5\}$



(a) Hiding set U with large size and measure.

(b) Smaller hiding set U^* with smaller measure.

Figure 6.5: Illustration of Example 6.29 (with vertices in hiding sets cross-marked).

in Figure 6.5(a). Then $U \cup W$ hides B, but U seems unnecessarily large. To get a better hiding set U^*, we can leave s_5 responsible for hiding v_5 but replace $\{v_1, u_2, u_3, v_3\}$ by $\{x_1, y_1\}$. The resulting set $U^* = \{x_1, y_1, s_5\}$ in Figure 6.5(b) has both smaller size and smaller measure (we leave the straightforward verification of this fact to the reader).

Intuitively, it seems that the configuration can be split into two disjoint components, namely $(B_1, W_1) = (\{x_1, y_1\}, \{w_3\})$ with hiding set $U_1 = \{v_1, u_2, u_3, v_3\}$ and $(B_2, W_2) = (\{v_5\}, \{s_6, s_7\})$ with hiding set $U_2 = \{s_5\}$, and that these two components are independent of one another. To improve the hiding set U, we need to do something locally about the bad hiding set U_1 in the first component, namely replace it with $U_1^* = \{x_1, y_1\}$, but we should keep the locally optimal hiding set U_2 in the second component.

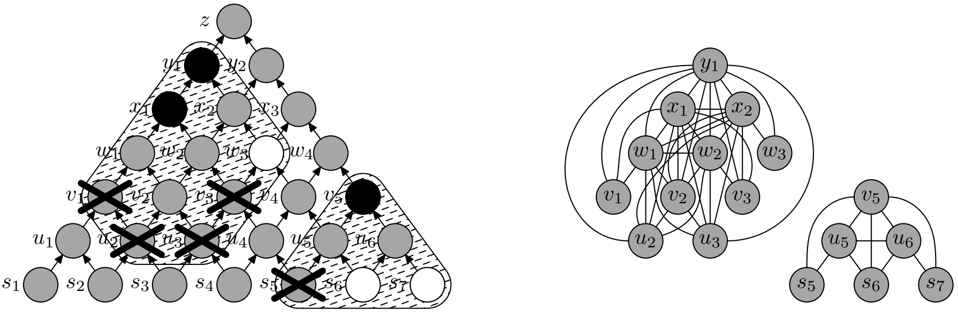
We want to formalize this understanding of how vertices in B, W and U depend on one another in a hiding set $U \cup W$ for B. The following definition constructs a graph that describes the structure of the hiding sets that we are studying in terms of these dependencies.

Definition 6.30 (Hiding set graph). For a tight (and non-empty) set of vertices X in G, the *hiding set graph* $\mathcal{H} = \mathcal{H}(G, X)$ is an undirected graph defined as follows:

- The set of vertices of $\mathcal{H}$ is $V(\mathcal{H}) = \lceil\!\lceil X \rceil\!\rceil$.
- The set of edges $E(\mathcal{H})$ of $\mathcal{H}$ consists of all pairs of vertices (x, y) for $x, y \in \lceil\!\lceil X \rceil\!\rceil$ such that $G^x_\Delta \cap \lceil\!\lceil X_{\lfloor\!\lfloor x \rfloor\!\rfloor} \rceil\!\rceil \cap G^y_\Delta \cap \lceil\!\lceil X_{\lfloor\!\lfloor y \rfloor\!\rfloor} \rceil\!\rceil \neq \emptyset$.

We say that the vertex set X is *hiding-connected* if $\mathcal{H}(G, X)$ is a connected graph.

When the graph G and vertex set X are clear from context, we will sometimes write only $\mathcal{H}(X)$ or even just $\mathcal{H}$. To illustrate Definition 6.30, we give an example.



(a) Vertices hidden by $U \cup W$.

(b) Hiding set graph $\mathcal{H}(U \cup W)$.

Figure 6.6: Pebbles with hiding set and corresponding hiding set graph.

Example 6.31. Consider again the configuration $(B, W) = (\{x_1, y_1, v_5\}, \{w_3, s_6, s_7\})$ from Example 6.29 with hiding set $U = \{v_1, u_2, u_3, v_3, s_5\}$, where we have shaded the set of hidden vertices in Figure 6.6(a). The hiding set graph $\mathcal{H}(X)$ for $X = U \cup W = \{v_1, u_2, u_3, v_3, w_3, s_5, s_6, s_7\}$ has been drawn in Figure 6.6(b). In accordance with the intuition sketched in Example 6.29, $\mathcal{H}(X)$ consists of two connected components.

Note that there are edges from the top vertex y_1 in the first component to every other vertex in this component and from the top vertex v_5 to every other vertex in the second component. We will prove presently that this is always the case (Lemma 6.32). Perhaps a more interesting edge in $\mathcal{H}(X)$ is, for instance, (w_1, x_2). This edge exists since $X\lfloor\!\lfloor w_1 \rfloor\!\rfloor = \{v_1, u_2, u_3\}$ and $X\lfloor\!\lfloor x_2 \rfloor\!\rfloor = \{u_2, u_3, v_3, w_3\}$ intersect and since as a consequence of this (which is easily verified) we have $\Pi^{w_1}_{\Delta} \cap \lceil\!\lceil X\lfloor\!\lfloor w_1 \rfloor\!\rfloor \rceil\!\rceil \cap \Pi^{x_2}_{\Delta} \cap \lceil\!\lceil X\lfloor\!\lfloor x_2 \rfloor\!\rfloor \rceil\!\rceil \neq \emptyset$. For the same reason, there is an edge (u_5, u_6) since $X\lfloor\!\lfloor u_5 \rfloor\!\rfloor = \{s_5, s_6\}$ and $X\lfloor\!\lfloor u_6 \rfloor\!\rfloor = \{s_6, s_7\}$ intersect.

Lemma 6.32. *Suppose for a tight vertex set X that $x \in \lceil\!\lceil X \rceil\!\rceil$ and $y \in X\lfloor\!\lfloor x \rfloor\!\rfloor$. Then x and y are in the same connected component of $\mathcal{H}(X)$.*

Proof. Note first that $x, y \in \lceil\!\lceil X \rceil\!\rceil$ by assumption, so x and y are both vertices in $\mathcal{H}(X)$. Since x is above y we have $G^x_\Delta \supseteq G^y_\Delta$ and we get $G^x_\Delta \cap \lceil\!\lceil X\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil \cap G^y_\Delta \cap \lceil\!\lceil X\lfloor\!\lfloor y \rfloor\!\rfloor \rceil\!\rceil = \lceil\!\lceil X\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil \cap G^y_\Delta \cap \{y\} = \{y\} \neq \emptyset$. Thus, (x, y) is an edge in $\mathcal{H}(X)$, so x and y are certainly in the same connected component. □

Corollary 6.33. *If X is tight and $x \in \lceil\!\lceil X \rceil\!\rceil$ then x and all of $X\lfloor\!\lfloor x \rfloor\!\rfloor$ are in the same connected component of $\mathcal{H}(X)$.*

The next lemma says that if $\mathcal{H}(X)$ is a hiding set graph with vertex set $V = \lceil\!\lceil X \rceil\!\rceil$, then the connected components $V_1, \ldots, V_k$ of $\mathcal{H}(X)$ are themselves hiding set graphs defined over the hiding-connected subsets $X \cap V_1, \ldots, X \cap V_k$.

Lemma 6.34 (Lemma 3.3 in [49]). *Let X be a tight set and let V_i be one of the connected components in $\mathcal{H}(X)$. Then the subgraph of $\mathcal{H}(X)$ induced by V_i is identical to the hiding set graph $\mathcal{H}(X \cap V_i)$ defined on the vertex subset $X \cap V_i$. In particular, it holds that $V_i = \llbracket X \cap V_i \rrbracket$.*

Proof. We need to show that $V_i = \llbracket X \cap V_i \rrbracket$ and that the edges of $\mathcal{H}(X)$ in V_i are exactly the edges in $\mathcal{H}(X \cap V_i)$. Let us first show that $y \in V_i$ if and only if $y \in \llbracket X \cap V_i \rrbracket$.

($\Rightarrow$) Suppose $y \in V_i$. Then $X_{\llbracket y \rrbracket} \subseteq V_i$ by Corollary 6.33. Also, $X_{\llbracket y \rrbracket} \subseteq X$ by definition, so $X_{\llbracket y \rrbracket} \subseteq X \cap V_i$. Since $y \in \llbracket X_{\llbracket y \rrbracket} \rrbracket$ by Lemma 6.27, clearly $y \in \llbracket X \cap V_i \rrbracket$.

($\Leftarrow$) Suppose $y \in \llbracket X \cap V_i \rrbracket$. Since X is tight, its subset $X \cap V_i$ must be tight as well. Applying Lemma 6.27 twice, we deduce that $(X \cap V_i)_{\llbracket y \rrbracket}$ hides y and that $X_{\llbracket y \rrbracket} \subseteq (X \cap V_i)_{\llbracket y \rrbracket}$ since $X_{\llbracket y \rrbracket}$ is contained in any subset of X that hides y. But then a third appeal to Lemma 6.27 yields that $(X \cap V_i)_{\llbracket y \rrbracket} \subseteq X_{\llbracket y \rrbracket}$ since $X_{\llbracket y \rrbracket} \subseteq (X \cap V_i)_{\llbracket y \rrbracket} \subseteq X \cap V_i$ and consequently

$$X_{\llbracket y \rrbracket} = (X \cap V_i)_{\llbracket y \rrbracket} \ . \tag{6.7}$$

By Corollary 6.33, y and all of $(X \cap V_i)_{\llbracket y \rrbracket} = X_{\llbracket y \rrbracket}$ are in the same connected component. Since $X_{\llbracket y \rrbracket} \subseteq V_i$ it follows that $y \in V_i$.

This shows that $V_i = \llbracket X \cap V_i \rrbracket$. Plugging (6.7) into Definition 6.30, we see that (x, y) is an edge in $\mathcal{H}(X)$ for $x, y \in V_i$ if and only if (x, y) is an edge in $\mathcal{H}(X \cap V_i)$. □

Now we are in a position to describe the structure of the proof that pyramid graphs have the LHC property.

Theorem 6.35 (Analogue of Theorem 3.7 in [49]). *Let $\mathbb{P} = (B, W)$ be any black-white pebble configuration on a pyramid Π. Then there is a vertex set U such that $U \cup W$ hides B, $\mathrm{pot}_\Pi(\mathbb{P}) = m(U)$ and either $U = B$ or $|U| < |B| + |W|$.*

The idea is to construct the graph $\mathcal{H} = \mathcal{H}(\Pi, U \cup W)$, study the different connected components in $\mathcal{H}$, find good hiding sets locally that satisfy the LHC property (which we prove is true for each local hiding-connected subset of $U \cup W$), and then add all of these partial hiding sets together to get a globally good hiding set.

Unfortunately, this does not quite work. Let us nevertheless attempt to do the proof, note where and why it fails, and then see how Klawe fixes the broken details.

Tentative proof of Theorem 6.35. Let U be a set of vertices in Π such that $U \cup W$ hides B and $\mathrm{pot}(\mathbb{P}) = m(U)$. Suppose that U has minimal size among all such sets, and furthermore that among all such minimum-measure and minimum-size sets U has the largest intersection with B.

Assume without loss of generality (Lemma 6.28) that $U \cup W$ is tight, so that we can construct $\mathcal{H}$. Let the connected components of $\mathcal{H}$ be $V_1, \dots, V_k$. For all

$i = 1, \ldots, k$, let $B_i = B \cap V_i$, $W_i = W \cap V_i$, and $U_i = U \cap V_i$. Lemma 6.34 says that $U_i \cup W_i$ hides B_i. In addition, all V_i are pairwise disjoint, so $|B| = \sum_{i=1}^{k} |B_i|$, $|W| = \sum_{i=1}^{k} |W_i|$ and $|U| = \sum_{i=1}^{k} |U_i|$.

Thus, if the LHC property 6.21 does not hold for U globally, there is some hiding-connected subset $U_i \cup W_i$ that hides B_i but for which $|U_i| \geq |B_i| + |W_i|$ and $U_i \neq B_i$. Note that this implies that $B_i \not\subseteq U_i$ since otherwise U_i would not be minimal.

Suppose that we would know that the LHC property is true for each connected component. Then we could find a vertex set U_i^* with $U_i^* \subseteq B_i$ or $\left|U_i^*\right| < |B_i| + |W_i|$ such that $U_i^* \cup W_i$ hides B_i and $m\left(U_i^*\right) \leq m(U_i)$. Setting $U^* = (U \setminus U_i) \cup U_i^*$, we would get a hiding set with either $|U^*| < |U|$ or $|U^* \cap B| > |U \cap B|$. The second inequality would hold since if $|U^*| = |U|$, then $\left|U_i^*\right| = |U_i| \geq |B_i \cup W_i|$ and this would imply $U_i^* = B_i$ and thus $\left|U_i^* \cap B_i\right| > |U_i \cap B_i|$. This would contradict how U was chosen above, and we would be home.

Almost. We would also need that U_i^* could be substituted for U_i in U without increasing the measure, i.e., that $m\left(U_i^*\right) \leq m\left(U_i\right)$ should imply $m\left((U \setminus U_i) \cup U_i^*\right) \leq m\left((U \setminus U_i) \cup U_i\right)$. And this turns out not to be true. □

The reason that the proof above does not quite work is that the measure in Definition 6.18 is ill-behaved with respect to unions. Klawe provides the following example of what can happen.

Example 6.36. With vertex labels as in Figures 6.2 and 6.4–6.6, let $X_1 = \{s_1, s_2\}$, $X_2 = \{w_1\}$ and $X_3 = \{s_3\}$. Then $m(X_1) = 4$ and $m(X_2) = 5$ but taking unions with X_3 we get that $m(X_1 \cup X_3) = 6$ and $m(X_2 \cup X_3) = 5$. Thus $m(X_1) < m(X_2)$ but $m(X_1 \cup X_3) > m(X_2 \cup X_3)$.

So it is not enough to show the LHC property locally for each connected component in the graph. We also need that sets U_i from different components can be combined into a global hiding set while maintaining measure inequalities. This leads to the following strengthened condition for connected components of $\mathcal{H}$.

Property 6.37 (Local limited hiding-cardinality property). We say that the pebble configuration $\mathbb{P} = (B, W)$ has the *Local limited hiding-cardinality property*, or just the *Local LHC property* for short, if for any vertex set U such that $U \cup W$ hides B and is hiding-connected, we can find a vertex set U^* such that

1. U^* is a hiding set for (B, W),
2. for any vertex set Y with $Y \cap U = \emptyset$ it holds that $m\left(Y \cup U^*\right) \leq m(Y \cup U)$,
3. $U^* \subseteq B$ or $\left|U^*\right| < |B| + |W|$.

We say that the graph G has the Local LHC property if all black-white pebble configurations $\mathbb{P} = (B, W)$ on G do.

Note that if the Local LHC property holds, this in particular implies that $m\big(U^*\big) \leq m(U)$ (just choose $Y = \emptyset$). Also, we immediately get that the LHC property holds globally.

Lemma 6.38. *If G has the Local limited hiding-cardinality property 6.37, then G has the Limited hiding-cardinality property 6.21.*

Proof. Consider the tentative proof of Theorem 6.35 and look at the point where it breaks down. If we instead use the Local LHC property to find U_i^*, this time we get that $m\big(U_i^*\big) \leq m\big(U_i\big)$ does indeed imply $m\big((U \setminus U_i) \cup U_i^*\big) \leq m\big((U \setminus U_i) \cup U_i\big)$, and the theorem follows. □

An obvious way to get the inequality $m(Y \cup U^*) \leq m(Y \cup U)$ in Property 6.37 would be to require that $m^j(U^*) \leq m^j(U)$ for all j, but we need to be slightly more general. The next definition identifies a sufficient condition for sets to behave well under unions with respect to the measure in Definition 6.18.

Definition 6.39. We write $U \precsim_m V$ if for all $j \geq 0$ there is an $i \leq j$ such that $m^j(U) \leq m^i(V)$.

Note that it is sufficient to verify the condition in Definition 6.39 for $j = 1, \ldots, \text{maxlevel}(U)$. For $j > \text{maxlevel}(U)$ we get $m^j(U) = 0$ and the inequality trivially holds.

It is immediate that $U \precsim_m V$ implies $m(U) \leq m(V)$, but the relation $\precsim_m$ gives us more information than that. Usual inequality $m(U) \leq m(V)$ holds if and only if for every j we can find an i such that $m^j(U) \leq m^i(V)$, but in the definition of $\precsim_m$ we are restricted to finding such an index i that is less than or equal to j. So not only is $m(U) \leq m(V)$ globally, but we can also explain locally at each level, by "looking downwards", why U has smaller measure than V.

In Example 6.36, $X_1 \not\precsim_m X_2$ since the relative cheapness of X_1 compared to X_2 is explained not by a lot of vertices in X_2 on low levels, but by one single high-level, and therefore expensive, vertex in X_2 which is far above X_1. This is why these sets behave badly under union. If we have two sets X_1 and X_2 with $X_1 \precsim_m X_2$, however, reversals of measure inequalities when taking unions as in Example 6.36 can no longer occur.

Lemma 6.40 (Lemma 3.4 in [49]). *If $U \precsim_m V$ and $Y \cap V = \emptyset$, then it holds that $m(Y \cup U) \leq m(Y \cup V)$.*

Proof. To show that $m(Y \cup U) \leq m(Y \cup V)$, we want to find for each level $j = 1, \ldots, \text{maxlevel}(Y \cup U)$ in U another level i in V such that $m^j(Y \cup U) \leq m^i(Y \cup V)$. We pick the $i \leq j$ provided by the definition of $U \precsim_m V$ such that $m^j(U) \leq m^i(V)$. Since $V \cap W = \emptyset$ and $i \leq j$ implies $Y\{\succeq j\} \subseteq Y\{\succeq i\}$, we get

$$\begin{aligned} m^j(Y \cup U) = j + 2 \cdot |(U \cup Y)\{\succeq j\}| \leq j + 2 \cdot |U\{\succeq j\}| + 2 \cdot |Y\{\succeq j\}| \leq \\ i + 2 \cdot |V\{\succeq i\}| + 2 \cdot |Y\{\succeq i\}| = m^i(Y \cup V) \quad (6.8) \end{aligned}$$

and the lemma follows. □

So when locally improving a blocking set U that does not satisfy the LHC property to some set U^* that does, if we can take care that $U^* \precsim_m U$ in the sense of Definition 6.39 we get the Local LHC property. All that remains is to show that this can indeed be done.

When "improving" U to U^*, we will strive to pick hiding sets of minimal size. The next definition makes this precise.

Definition 6.41. For any set of vertices X, let

$$L_{\succeq j}(X) = \min\{|Y| : X\{\succeq j\} \subseteq \llbracket Y \rrbracket \text{ and } Y\{\succeq j\} = Y\}$$

denote the size of a smallest set Y such that all vertices in Y are on level j or higher and Y hides all vertices in X on level j or higher.

Note that we only require of Y to hide $X\{\succeq j\}$ and not all of X. Given the condition that $Y = Y\{\succeq j\}$, this set cannot hide any vertices in $X\{\prec j\}$. We make a few easy observations.

Observation 6.42. *Suppose that X is a set of vertices in a layered graph G. Then:*

1. *$L_{\succeq 0}(X)$ is the minimal size of any hiding set for X.*
2. *If $X \subseteq Y$, then $L_{\succeq j}(X) \leq L_{\succeq j}(Y)$ for all j.*
3. *It always holds that $L_{\succeq j}(X) \leq |X\{\succeq j\}| \leq |X|$.*

Proof. Part 1 follows from the fact that $V\{\succeq 0\} = V$ for any set V. If $X \subseteq Y$, then $X\{\succeq j\} \subseteq Y\{\succeq j\}$ and any hiding set for $X\{\succeq j\}$ works also for $Y\{\succeq j\}$, which yields part 2. Part 3 holds since $X\{\succeq j\} \subseteq X$ is always a possible hiding set for itself. □

For any vertex set V in any layered graph G, we can always find a set hiding V that has "minimal cardinality at each level" in the sense of Definition 6.41.

Lemma 6.43 (Lemma 3.5 in [49]). *For any vertex set V we can find a hiding set V^* such that $\left|V^*\{\succeq j\}\right| \leq L_{\succeq j}(V)$ for all j, and either $V^* = V$ or $|V^*| < |V|$.*

Proof. If $|V\{\succeq j\}| \leq L_{\succeq j}(V)$ for all j, we can choose $V^* = V$. Suppose this is not the case, and let k be minimal such that $|V\{\succeq k\}| > L_{\succeq k}(V)$. Let V' be a minimum-size hiding set for $V\{\succeq k\}$ with $V' = V'\{\succeq k\}$ and $\left|V'\right| = |L_{\succeq k}(V)|$ and set $V^* = V\{\prec k\} \mathbin{\dot\cup} V'$. Since $V\{\prec k\}$ hides itself (any set does), we have that V^* hides $V = V\{\prec k\} \mathbin{\dot\cup} V\{\succeq k\}$ and that

$$\left|V^*\right| = |V\{\prec k\}| + |V'| < |V\{\prec k\}| + |V\{\succeq k\}| = |V| \ . \tag{6.9}$$

Combining (6.9) with part 1 of Observation 6.42, we see that the minimal index found above must be $k = 0$. Going through the same argument as above again, we see that $\left|V^*\{\succeq j\}\right| \leq L_{\succeq j}(V)$ for all j, since otherwise (6.9) would yield a contradiction to the fact that $V' = V'\{\succeq 0\}$ was chosen as a minimum-size hiding set for V. □

We noted above that $L_{\succeq 0}(X)$ is the cardinality of a minimum-size hiding set of X. For $j > 0$, the quantity $L_{\succeq j}(X)$ is large if one needs many vertices on level $\geq j$ to hide $X\{\succeq j\}$, i.e., if $X\{\succeq j\}$ is "spread out" in some sense. Let us consider a pyramid graph and suppose that X is a tight and hiding-connected set in which the level-difference $\mathrm{maxlevel}(X) - \mathrm{minlevel}(X)$ is large. Then it seems that $|X|$ should also have to be large, since the pyramid "fans out" so quickly. This intuition might be helpful when looking at the next, crucial definition of Klawe.

Definition 6.44 (Spreading graph). We say that the layered DAG G is a *spreading graph* if for every (non-empty) hiding-connected set X in G and every level $j = 1, \ldots, \mathrm{maxlevel}(\lceil\!\lceil X \rceil\!\rceil)$, the *spreading inequality*

$$|X| \geq L_{\succeq j}(\lceil\!\lceil X \rceil\!\rceil) + j - \mathrm{minlevel}(X) \tag{6.10}$$

holds.

Let us try to give some more intuition for Definition 6.44 by considering two extreme cases in a pyramid graph:

- For $j \leq \mathrm{minlevel}(X)$, we have that the term $j - \mathrm{minlevel}(X)$ is non-positive, $X\{\succeq j\} = X$, and $\lceil\!\lceil X\{\succeq j\} \rceil\!\rceil = \lceil\!\lceil X \rceil\!\rceil$. In this case, (6.10) is just the trivial fact that no set that hides $\lceil\!\lceil X \rceil\!\rceil$ need be larger than X itself.
- Consider $j = \mathrm{maxlevel}(\lceil\!\lceil X \rceil\!\rceil)$, and suppose that $\lceil\!\lceil X\{\succeq j\} \rceil\!\rceil$ is a single vertex v with $X_{\lfloor\!\lfloor x \rfloor\!\rfloor} = X$. Then (6.10) requires that $|X| \geq 1 + \mathrm{level}(x) - \mathrm{minlevel}(X)$, and this can be proven to hold by the "converging paths" argument of Theorem 6.10 and Observation 6.12.

Very loosely, Definition 6.44 says that if X contains vertices at low levels that help to hide other vertices at high levels, then X must be a large set. Just as we tried to argue above, the spreading inequality (6.10) does indeed hold for pyramids.

Theorem 6.45 ([49]). *Pyramids are spreading graphs.*

Unfortunately, the proof of Theorem 6.45 in [49] is rather involved. The analysis is divided into two parts, by first showing that a class of so-called *nice graphs* are spreading, and then demonstrating that pyramid graphs are nice. In Section 6.5.2, we give a simplified, direct proof of the fact that pyramids are spreading that might be of independent interest.

Accepting Theorem 6.45 on faith for now, we are ready for the decisive lemma: If our layered DAG is a spreading graph and if $U \cup W$ is a hiding-connected set hiding B such that U is too large for the conditions in the Local limited hiding-cardinality property 6.37 to hold, then replacing U by the minimum-size hiding set in Lemma 6.43 we get a hiding set in accordance with the Local LHC property.

Lemma 6.46 (Lemma 3.6 in [49]). *Suppose that B, W, U are vertex sets in a layered spreading graph G such that $U \cup W$ hides B and is tight and hiding-connected. Then there is a vertex set U^* such that $U^* \cup W$ hides B, $U^* \precsim_m U$, and either $U^* = B$ or $|U^*| < |B| + |W|$.*

Postponing the proof of Lemma 6.46 for a moment, let us note that if we combine this lemma with Lemma 6.40 and Theorem 6.45, the Local limited hiding-cardinality property for pyramids follows.

Corollary 6.47. *Pyramid graphs have the Local limited hiding-cardinality property 6.37.*

Proof of Corollary 6.47. This is more or less immediate, but we write down the details for completeness. Since pyramids are spreading by Theorem 6.45, Lemma 6.46 says that U^* is a hiding set for (B, W) and that $U^* \precsim_m U$. Lemma 6.40 then yields that $m(Y \cup U^*) \leq m(Y \cup U)$ for all Y with $Y \cap U = \emptyset$. Finally, Lemma 6.46 also tells us that $U^* \subseteq B$ or $|U^*| < |B| + |W|$, and thus all conditions in Property 6.37 are satisfied. □

Continuing by plugging Corollary 6.47 into Lemma 6.38, we get the global LHC property in Theorem 6.35 on page 76. So all that is needed to conclude Klawe's proof of the lower bound for the black-white pebbling price of pyramids is to prove Theorem 6.45 and Lemma 6.46. We attend to Lemma 6.46 right away, deferring a proof of Theorem 6.45 to the next subsection.

Proof of Lemma 6.46. If $|U| < |B| + |W|$ we can pick $U^* = U$ and be done, so suppose that $|U| \geq |B| + |W|$. Intuitively, this should mean that U is unnecessarily large, so it ought to be possible to do better. In fact, U is so large that we can just ignore W and pick a better U^* that hides B all on its own.

Namely, let U^* be a minimum-size hiding set for B as in Lemma 6.43. Then either $U^* = B$ or $\left|U^*\right| < |B| \leq |B| + |W|$. To prove the lemma, we also need to show that $U^* \precsim_m U$, which will guarantee that U^* behaves well under union with other sets with respect to measure.

Before we do the the formal calculations, let us try to provide some intuition for why it should be the case that $U^* \precsim_m U$ holds, i.e., that for every j we can find an $i \leq j$ such that $m^j\big(U^*\big) \leq m^i(U)$. Perhaps it will be helpful at this point for the reader to look at Example 6.29 again, where the replacement of $U_1 = \{v_1, u_2, u_3, v_3\}$ in Figure 6.5(a) by $U_1^* = \{x_1, y_1\}$ in Figure 6.5(b) shows Lemmas 6.43 and 6.46 in action.

Suppose first that $j \leq \text{minlevel}(U \cup W) \leq \text{minlevel}(U)$. Then the measure inequality $m^j(U^*) \leq m^j(U)$ is obvious, since $U\{\succeq j\} = U$ is so large that it can easily pay for all of U^*, let alone $U^*\{\succeq j\} \subseteq U^*$.

For $j > \text{minlevel}(U \cup W)$, however, we can worry that although our hiding set U^* does indeed have small size, the vertices in U^* might be located on high levels in the graph and be very expensive since they were chosen without regard to measure. Just throwing away all white pebbles and picking a new set U^* that hides B on its own is quite a drastic move, and it is not hard to construct examples where this is very bad in terms of potential (say, exchanging s_5 for v_5 in the hiding set of Example 6.29). The reason that this nevertheless works is that $|U|$ is so large, that, in addition, $U \cup W$ is hiding-connected, and that, finally, the graph under consideration is spreading. Thanks to this, if there are a lot of expensive vertices in $U^*\{\succeq j\}$ on or above some high level j resulting in a large partial measure $m^j\bigl(U^*\bigr)$, the number of vertices on or above level $L = \mathrm{minlevel}(U \cup W)$ in $U = U\{\succeq L\}$ is large enough to yield at least as large a partial measure $m^L\bigl(U\bigr)$.

Let us do the formal proof, divided into the two cases above.

1. $j \leq \mathrm{minlevel}(U \cup W)$: Using the lower bound on the size of U and that level j is no higher than the minimal level of U, we get

$$\begin{aligned}
m^j\bigl(U^*\bigr) &= j + 2 \cdot \bigl|U^*\{\succeq j\}\bigr| && \bigl[\text{ by definition of } m^j(\cdot) \text{ }\bigr] \\
&\leq j + 2 \cdot \bigl|U^*\bigr| && \bigl[\text{ since } V\{\succeq j\} \subseteq V \text{ for any } V \text{ }\bigr] \\
&\leq j + 2 \cdot |B| && \bigl[\text{ by construction of } U^* \text{ in Lemma 6.43 }\bigr] \\
&\leq j + 2 \cdot |U| && \bigl[\text{ by assumption } |U| \geq |B| + |W| \geq |B| \text{ }\bigr] \\
&= j + 2 \cdot \bigl|U\{\succeq j\}\bigr| && \bigl[\text{ } U\{\succeq j\} = U \text{ since } j \leq \mathrm{minlevel}(U) \text{ }\bigr] \\
&= m^j(U) && \bigl[\text{ by definition of } m^j(\cdot) \text{ }\bigr]
\end{aligned}$$

and we can choose $i = j$ in Definition 6.39.

2. $j > \mathrm{minlevel}(U \cup W)$: Let $L = \mathrm{minlevel}(U \cup W)$. The black pebbles in B are hidden by $U \cup W$, or in formal notation $B \subseteq \llbracket U \cup W \rrbracket$, so

$$L_{\succeq j}(B) \leq L_{\succeq j}\bigl(\llbracket U \cup W \rrbracket\bigr) \tag{6.11}$$

holds by part 2 of Observation 6.42. Moreover, $U \cup W$ is a hiding-connected set of vertices in a spreading graph G, so the spreading inequality in Definition 6.44 says that $|U \cup W| \geq L_{\succeq j}\bigl(\llbracket U \cup W \rrbracket\bigr) + j - L$, or

$$j + L_{\succeq j}\bigl(\llbracket U \cup W \rrbracket\bigr) \leq L + |U \cup W| \tag{6.12}$$

after reordering. Combining (6.11) and (6.12) we have that

$$j + L_{\succeq j}(B) \leq L + |U \cup W| \tag{6.13}$$

and it follows that

$$\begin{aligned}
m^j(U^*) &= j + 2 \cdot \bigl|U^*\{\succeq j\}\bigr| && \bigl[\text{ by definition of } m^j(\cdot) \text{ }\bigr] \\
&\leq j + \bigl|U^*\{\succeq j\}\bigr| + \bigl|U^*\bigr| && \bigl[\text{ since } V\{\succeq j\} \subseteq V \text{ for any } V \text{ }\bigr] \\
&\leq j + L_{\succeq j}(B) + |B| && \bigl[\text{ by construction of } U^* \text{ in Lemma 6.43 }\bigr] \\
&\leq L + |U \cup W| + |B| && \bigl[\text{ by the inequality (6.13) }\bigr] \\
&\leq L + 2 \cdot |U| && \bigl[\text{ by assumption } |U| \geq |B| + |W| \text{ }\bigr] \\
&= L + 2 \cdot |U\{\succeq L\}| && \bigl[\text{ } U\{\succeq L\} = U \text{ since } L \leq \mathrm{minlevel}(U) \text{ }\bigr] \\
&= m^L(U) && \bigl[\text{ by definition of } m^L(\cdot) \text{ }\bigr]
\end{aligned}$$

Thus, the partial measure of U at the minimum level L is always larger than the partial measure of U^* at levels j above this minimum level, and we can choose $i = L$ in Definition 6.39.

Consequently, $U^* \precsim_m U$, and the lemma follows. □

Concluding this subsection, we want to make a comment about Lemmas 6.43 and 6.46 and try to rephrase what they say about hiding sets. Given a tight set $U \cup W$ such that $B \subseteq \lceil\!\lceil U \cup W \rceil\!\rceil$, we can always pick a U^* as in Lemma 6.43 with $U^* = B$ or $\left|U^*\right| < |B|$ and with $\left|U^*\{\succeq j\}\right| \leq L_{\succeq j}(B)$ for all j. This will sometimes be a good idea, and sometimes not. Just as in Lemma 6.46, for $j >$ minlevel$(U \cup W)$ we can always prove that

$$m^j(U^*) \leq \mathrm{minlevel}(U \cup W) + |U| + (|B| + |W|) \ . \tag{6.14}$$

The key message of Lemma 6.46 is that replacing U by U^* is a good idea if U is sufficiently large, namely if $|U| \geq |B| + |W|$, in which case we are guaranteed to get $m^j(U^*) \leq m^L(U)$ for $L = \mathrm{minlevel}(U \cup W)$.

6.5.2 Pyramids Are Spreading Graphs

The fact that pyramids are spreading graphs, that is, that they satisfy the inequality (6.10), is a consequence of the following lemma.

Lemma 6.48 (Ice-Cream Cone Lemma). *If X is a tight vertex set in a pyramid Π such that $\mathcal{H}(X)$ is a connected graph with vertex set $V = \lceil\!\lceil X \rceil\!\rceil$, then there is a unique vertex $x \in V$ such that $X = X_{\lfloor\!\lfloor x \rfloor\!\rfloor}$ and $V = \lceil\!\lceil X_{\lfloor\!\lfloor x \rfloor\!\rfloor} \rceil\!\rceil \subseteq \Pi^x_\Delta$.*

What the lemma says it that for any tight vertex set X, the connected components $V_1, \ldots, V_k$ look like ragged ice-cream cones turned upside down. Moreover, for each "ice-cream cone" V_i, all vertices in $X \cap V_i$ are needed to hide the top vertex. The two connected components in Figure 6.6 are both examples of such "ice-cream cones."

Before proving Lemma 6.48, we show how this lemma can be used to establish that pyramid graphs are spreading by a converging-paths argument as in Observation 6.12.

Proof of Theorem 6.45. Suppose that X is a tight and hiding-connected set, i.e., such that $\mathcal{H}(X)$ is a single connected component with set of vertices $V = \lceil\!\lceil X \rceil\!\rceil$. Let $x \in V$ be the vertex given by Lemma 6.48 such that $X = X_{\lfloor\!\lfloor x \rfloor\!\rfloor}$ and $V = \lceil\!\lceil X_{\lfloor\!\lfloor x \rfloor\!\rfloor} \rceil\!\rceil \subseteq \Pi^x_\Delta$, and let $M = \mathrm{level}(x)$.

For any $j \leq M$ we have

$$L_{\succeq j}(\lceil\!\lceil X \rceil\!\rceil) \leq M - j + 1 \ . \tag{6.15}$$

This is so since there are only so many vertices on level j in Π^x_Δ and the set of all these vertices must hide everything in $\lceil\!\lceil X \rceil\!\rceil$ above level j since $\lceil\!\lceil X \rceil\!\rceil \subseteq \Pi^x_\Delta$.

By assumption X is tight and all of X is needed to hide x, i.e., $X = X_{\lfloor\!\lfloor x \rfloor\!\rfloor}$. Pick a vertex $v \in X$ on bottom level $L = \mathrm{minlevel}(X)$. Since $v \in X_{\lfloor\!\lfloor x \rfloor\!\rfloor}$ there is a path $P : v \rightsquigarrow x$ such that $P \cap X = \{v\}$. Consider the set of converging source paths for P in Observation 6.12. All these converging paths $P_1, P_2, \ldots, P_{M-L}$ must be blocked by distinct vertices in $X \setminus \{v\}$, since $P_i \cap P_j \subseteq P \setminus \{v\}$ and $P \setminus \{v\}$ does not intersect X. From this the inequality

$$|X| \geq M - L + 1 \tag{6.16}$$

follows. By combining (6.15) and (6.16), we get that

$$|X| - L_{\succeq j}(\lceil\!\lceil X \rceil\!\rceil) \geq M - L + 1 - (M - j + 1) = j - L \tag{6.17}$$

which is the required spreading inequality (6.10). □

The rest of this subsection is devoted to proving the Ice-Cream Cone Lemma. We will use that fact that pyramids are planar graphs where we can talk about left and right. More precisely, the following (immediate) observation will be central in our proof.

Observation 6.49. *Suppose for a planar DAG G that we have a source path P to a vertex w and two vertices $u, v \in G^{w}_{\Delta}$ on opposite sides of P. Then any path $Q : u \rightsquigarrow v$ must intersect P.*

Given a vertex v in a pyramid Π, there is a unique path that passes through v and in every vertex u moves to the right-hand successor of u. We will refer to this path as the *north-east path* through v, or just the *NE-path* through v for short, and denote it by $P_{\mathrm{NE}}(v)$. The path through v always moving to the left is the *north-west path* or *NW-path* through v, and is denoted $P_{\mathrm{NW}}(v)$. For instance, for the vertex v_4 in our running example pyramid in Figure 6.2 we have $P_{\mathrm{NE}}(v_4) = \{s_4, u_4, v_4, w_4\}$ and $P_{\mathrm{NW}}(v_4) = \{s_6, u_5, v_4, w_3, x_2, y_1\}$. To simplify the proofs in what follows, we make a couple of observations.

Observation 6.50. *Suppose that X is a tight set of vertices in a pyramid Π and that $v \in \lceil\!\lceil X \rceil\!\rceil$. Then $\lceil\!\lceil X_{\lfloor\!\lfloor v \rfloor\!\rfloor} \rceil\!\rceil \subseteq \Pi^{v}_{\Delta}$.*

Proof. Since all vertices in $X_{\lfloor\!\lfloor v \rfloor\!\rfloor}$ have a path to v by definition, it holds that $X_{\lfloor\!\lfloor v \rfloor\!\rfloor} \subseteq \Pi^{v}_{\Delta}$. Any vertex $u \in \Pi \setminus \Pi^{v}_{\Delta}$ must lie either to the left of $P_{\mathrm{NE}}(v)$ or to the right of $P_{\mathrm{NW}}(v)$ (or both). In the first case, $P_{\mathrm{NE}}(u)$ is a path via u that does not intersect $X_{\lfloor\!\lfloor v \rfloor\!\rfloor}$, so $u \notin \lceil\!\lceil X_{\lfloor\!\lfloor v \rfloor\!\rfloor} \rceil\!\rceil$. In the second case, we can draw the same conclusion by looking at $P_{\mathrm{NW}}(u)$. Thus, $\bigl(\Pi \setminus \Pi^{v}_{\Delta}\bigr) \cap \lceil\!\lceil X_{\lfloor\!\lfloor v \rfloor\!\rfloor} \rceil\!\rceil = \emptyset$. □

Observation 6.51. *Suppose that X is a tight set of vertices in a DAG G and that $v \in \lceil\!\lceil X \rceil\!\rceil$. Then there is a source path P to v such that $|P \cap X| = 1$.*

Proof. Let P_1 be any source path to v and note that P_1 intersects X since $v \in \lceil\!\lceil X \rceil\!\rceil$. Let y be the last vertex on P_1 in $P_1 \cap X$, i.e., the vertex on the highest level in this intersection. Since X is tight, there is a source path P_2 to y that does not intersect $X \setminus \{y\}$. Let P be the path that starts like P_2 and then switches to P_1 in y. Then $|P \cap X| = |\{y\}| = 1$. □

Using Observations 6.50 and 6.51, we can simplify the definition of the hiding set graph. Note that Observation 6.50 is not true for arbitrary layered DAGs, however, or even for arbitrary layered planar DAGs, so the simplification below does not work in general.

Proposition 6.52. *Let $\mathcal{H} = \mathcal{H}(\Pi, X)$ be the hiding set graph for a tight set of vertices X in a pyramid Π, and suppose that $u, v \in \lceil\!\lceil X \rceil\!\rceil$. Then the following conditions are equivalent:*

1. *(u, v) is an edge in $\mathcal{H}$, i.e., $\Pi^u_\Delta \cap \lceil\!\lceil X\lfloor\!\lfloor u \rfloor\!\rfloor \rceil\!\rceil \cap \Pi^v_\Delta \cap \lceil\!\lceil X\lfloor\!\lfloor v \rfloor\!\rfloor \rceil\!\rceil \neq \emptyset$.*
2. *$\lceil\!\lceil X\lfloor\!\lfloor u \rfloor\!\rfloor \rceil\!\rceil \cap \lceil\!\lceil X\lfloor\!\lfloor v \rfloor\!\rfloor \rceil\!\rceil \neq \emptyset$.*
3. *$X\lfloor\!\lfloor u \rfloor\!\rfloor \cap X\lfloor\!\lfloor v \rfloor\!\rfloor \neq \emptyset$.*

Proof. The directions (1) $\Rightarrow$ (2) and (3) $\Rightarrow$ (2) are immediate. The implication (2) $\Rightarrow$ (1) also follows easily, since $\lceil\!\lceil X\lfloor\!\lfloor u \rfloor\!\rfloor \rceil\!\rceil \subseteq \Pi^u_\Delta$ and $\lceil\!\lceil X\lfloor\!\lfloor v \rfloor\!\rfloor \rceil\!\rceil \subseteq \Pi^v_\Delta$ by Observation 6.50. To prove (2) $\Rightarrow$ (3), fix some vertex $w \in \lceil\!\lceil X\lfloor\!\lfloor u \rfloor\!\rfloor \rceil\!\rceil \cap \lceil\!\lceil X\lfloor\!\lfloor v \rfloor\!\rfloor \rceil\!\rceil$ and let P be a source path to w as in Observation 6.51 with $P \cap X = \{y\}$ for some vertex y. Since $P \cap X\lfloor\!\lfloor u \rfloor\!\rfloor \neq \emptyset \neq P \cap X\lfloor\!\lfloor u \rfloor\!\rfloor$ by assumption, we have $y \in X\lfloor\!\lfloor u \rfloor\!\rfloor \cap X\lfloor\!\lfloor v \rfloor\!\rfloor \neq \emptyset$. □

As the first part of the proof of Lemma 6.48, we show that all vertices hidden by a hiding-connected set X are contained in a subpyramid, the top vertex of which is also hidden by X. This gives the ice-cream cone shape alluded to by the name of the lemma.

Lemma 6.53. *Let $\mathcal{H} = \mathcal{H}(\Pi, X)$ be the hiding set graph of a hiding-connected vertex set X in a pyramid Π. Then there is a unique vertex $x \in \lceil\!\lceil X \rceil\!\rceil$ such that $\lceil\!\lceil X \rceil\!\rceil \subseteq \Pi^x_\Delta$.*

Proof. It is clear that at most one vertex $x \in \lceil\!\lceil X \rceil\!\rceil$ can have the properties stated in the lemma. We show that such a vertex exists. As a quick preview of the proof, we note that it is easy to find a unique vertex x on minimal level such that $\lceil\!\lceil X \rceil\!\rceil \subseteq \Pi^x_\Delta$. The crucial part of the lemma is that x is hidden by X. The reason that this holds is that the graph $\mathcal{H}$ is connected. If $x \notin \lceil\!\lceil X \rceil\!\rceil$, we can find a source path P to the top vertex z of the pyramid such that P does not intersect X but there are vertices in $\mathcal{H}$ both to the left and to the right of P. But there is no way we can have an edge crossing P in $\mathcal{H}$, so the hiding set graph cannot be connected after all. Contradiction.

The above paragraph really is the whole proof, but let us also provide the (somewhat tedious) formal details for completeness. To follow the formalization of the argument, the reader might be helped by looking at Figure 6.7. Suppose that Π has height h and let $s_1, s_2, \ldots, s_{h+1}$ be the sources enumerated from left to right. Look at the north-east paths $P_{\mathrm{NE}}(s_1), P_{\mathrm{NE}}(s_2), \ldots$ and let s_i be the first vertex such that $P_{\mathrm{NE}}(s_i) \cap \lceil\!\lceil X \rceil\!\rceil \neq \emptyset$. Similarly, consider $P_{\mathrm{NW}}(s_{h+1}), P_{\mathrm{NW}}(s_h), \ldots$ and let s_j be the first vertex such that $P_{\mathrm{NW}}(s_j) \cap \lceil\!\lceil X \rceil\!\rceil \neq \emptyset$. It clearly holds that $i \leq j$.

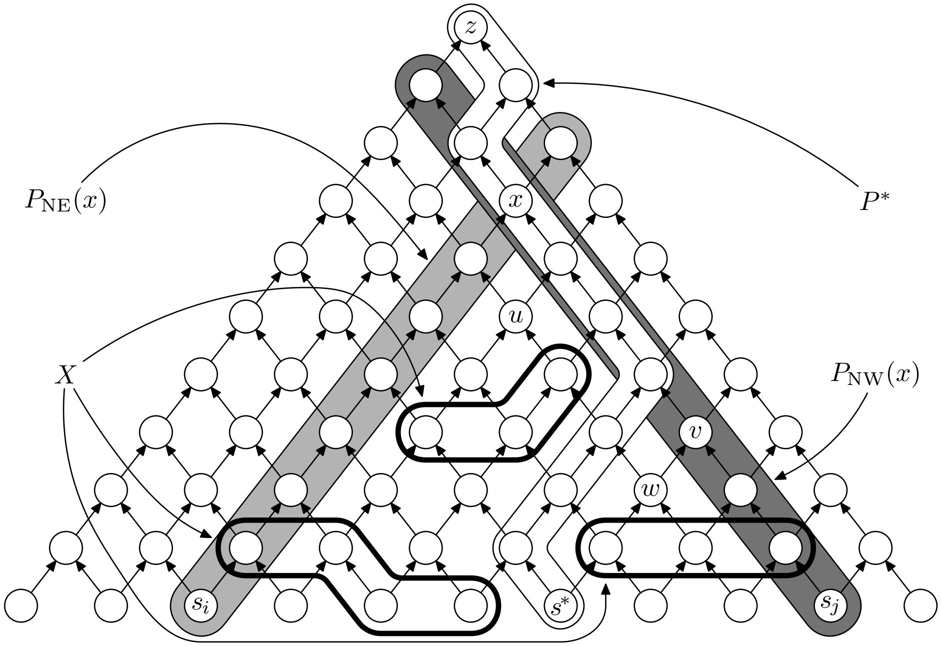


Figure 6.7: Illustration of proof of Lemma 6.53 that $\mathcal{H}$ is not connected if $x \notin \lceil\!\lceil X \rceil\!\rceil$.

Let x be the unique vertex where $P_{\mathrm{NE}}(s_i)$ and $P_{\mathrm{NW}}(s_j)$ intersect. By construction, we have $\lceil\!\lceil X \rceil\!\rceil \subseteq \Pi^x_\Delta$, since no NE-path to the left of $P_{\mathrm{NE}}(s_i) = P_{\mathrm{NE}}(x)$ intersects $\lceil\!\lceil X \rceil\!\rceil$ and neither does any NW-path to the right of $P_{\mathrm{NW}}(s_j) = P_{\mathrm{NW}}(x)$. We need to show that it also holds that $x \in \lceil\!\lceil X \rceil\!\rceil$.

To derive a contradiction, suppose instead that $x \notin \lceil\!\lceil X \rceil\!\rceil$. By definition, there is a path P from some source s^* to x such that $P \cap \lceil\!\lceil X \rceil\!\rceil = \emptyset$. P cannot coincide with $P_{\mathrm{NE}}(x)$ or $P_{\mathrm{NW}}(x)$ since the latter two paths both intersect $\lceil\!\lceil X \rceil\!\rceil$ by construction. Since $\Pi^{\triangledown}_{x} \cap \lceil\!\lceil X \rceil\!\rceil = \emptyset$, we can extend P to a path $P^* : s^* \rightsquigarrow z$ via x having the property that $P^* \cap \lceil\!\lceil X \rceil\!\rceil = \emptyset$ but there are vertices in $\mathcal{H}(X)$ both to the left and to the right of P^*, namely, the non-empty sets $P_{\mathrm{NE}}(x) \cap \lceil\!\lceil X \rceil\!\rceil \cap \Pi^x_\Delta$ and $P_{\mathrm{NW}}(x) \cap \lceil\!\lceil X \rceil\!\rceil \cap \Pi^x_\Delta$. We claim that this implies that $\mathcal{H}$ is not connected. This is a contradiction to the assumptions in the statement of the lemma and it follows that $x \in \lceil\!\lceil X \rceil\!\rceil$ must hold.

To establish the claim, note that if $\mathcal{H}$ is connected, there must exist some edge (u, v) between a vertex u to the left of P^* and a vertex v to the right of P^*. Then Proposition 6.52 says that $\lceil\!\lceil X_{\lfloor\!\lfloor u \rfloor\!\rfloor} \rceil\!\rceil \cap \lceil\!\lceil X_{\lfloor\!\lfloor v \rfloor\!\rfloor} \rceil\!\rceil \neq \emptyset$. Pick any vertex $w \in \lceil\!\lceil X_{\lfloor\!\lfloor u \rfloor\!\rfloor} \rceil\!\rceil \cap \lceil\!\lceil X_{\lfloor\!\lfloor v \rfloor\!\rfloor} \rceil\!\rceil$ and assume without loss of generality that w is on the right-hand side of P^*. We prove that such a vertex w cannot exist. See the example vertices labelled u, v and w in Figure 6.7, which illustrate the fact that $w \notin \lceil\!\lceil X_{\lfloor\!\lfloor u \rfloor\!\rfloor} \rceil\!\rceil$ if $w \in \lceil\!\lceil X_{\lfloor\!\lfloor v \rfloor\!\rfloor} \rceil\!\rceil$.

Since w is assumed to be hidden by $\lceil\!\lceil X\lfloor\!\lfloor u \rfloor\!\rfloor \rceil\!\rceil$, the NW-path through w must intersect $X\lfloor\!\lfloor u \rfloor\!\rfloor$ somewhere before w or in w. Fix any $y \in P_{\mathrm{NW}}(w) \cap X\lfloor\!\lfloor u \rfloor\!\rfloor \cap \Pi^{w}_{\triangle}$ and note that y must also be located to the right of P^*. By Definition 6.26, there is a source path P' via y to u such that $P' \cap X = \{y\}$. But P' must intersect P^* somewhere above y, since y is to the right and u is to the left of P^*. (Here we use Observation 6.49.) Consider the source path that starts like P^* and then switches to P' at some intersection point in $P' \cap P^* \cap \Pi^{\triangledown}_{y}$. This path reaches u but does not intersect X, contradicting the assumption $u \in \lceil\!\lceil X \rceil\!\rceil$. It follows that $\lceil\!\lceil X\lfloor\!\lfloor u \rfloor\!\rfloor \rceil\!\rceil \cap \lceil\!\lceil X\lfloor\!\lfloor v \rfloor\!\rfloor \rceil\!\rceil = \emptyset$ for all u and v on different sides of P^*, so there are no edges across P^* in $\mathcal{H}$. This proves the claim. □

The second part needed to prove Lemma 6.48 is that all vertices in X are required to hide the top vertex $x \in \lceil\!\lceil X \rceil\!\rceil$ found in Lemma 6.53.

Lemma 6.54. *Let $\mathcal{H} = \mathcal{H}(\Pi, X)$ be the hiding set graph of a hiding-connected vertex set X in a pyramid Π and let $x \in \lceil\!\lceil X \rceil\!\rceil$ be the unique vertex such that $\lceil\!\lceil X \rceil\!\rceil \subseteq \Pi^{x}_{\triangle}$. Then $X = X\lfloor\!\lfloor x \rfloor\!\rfloor$.*

Proof. By definition, $X\lfloor\!\lfloor x \rfloor\!\rfloor \subseteq X$. We want to show that $X\lfloor\!\lfloor x \rfloor\!\rfloor = X$. Again, let us first try to convey some intuition why the lemma is true. If $X \setminus X\lfloor\!\lfloor x \rfloor\!\rfloor \neq \emptyset$, since X is hiding-connected there must exist some vertex hidden by all of X but not by just $X\lfloor\!\lfloor x \rfloor\!\rfloor$ or $X \setminus X\lfloor\!\lfloor x \rfloor\!\rfloor$ (otherwise there can be no edge between the components of $\mathcal{H}$ containing $X\lfloor\!\lfloor x \rfloor\!\rfloor$ and $X \setminus X\lfloor\!\lfloor x \rfloor\!\rfloor$, respectively). But if so, it can be shown that the extra vertices in $X \setminus X\lfloor\!\lfloor x \rfloor\!\rfloor$ help $X\lfloor\!\lfloor x \rfloor\!\rfloor$ to hide one of its own vertices. This contradicts the fact that X is tight, so we must have $X\lfloor\!\lfloor x \rfloor\!\rfloor = X$ which proves the lemma.

Let us fill in the formal details in this proof sketch. Assume, to derive a contradiction, that $X\lfloor\!\lfloor x \rfloor\!\rfloor \neq X$. Since X is tight, it holds that $(X \setminus X\lfloor\!\lfloor x \rfloor\!\rfloor) \cap \lceil\!\lceil X\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil = \emptyset$, so $\mathcal{H}$ contains vertices outside of $\lceil\!\lceil X\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil$. Since $\mathcal{H}$ is connected, there must exist some edge (u, u') between a pair of vertices $u \in \lceil\!\lceil X \rceil\!\rceil \setminus \lceil\!\lceil X\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil$ and $u' \in \lceil\!\lceil X\lfloor\!\lfloor x \rfloor\!\rfloor \rceil\!\rceil$. Lemma 6.27 says that $X\lfloor\!\lfloor u' \rfloor\!\rfloor \subseteq X\lfloor\!\lfloor x \rfloor\!\rfloor$ and Proposition 6.52 then tells us that $X\lfloor\!\lfloor u \rfloor\!\rfloor \cap X\lfloor\!\lfloor x \rfloor\!\rfloor \neq \emptyset$. Also, $X\lfloor\!\lfloor u \rfloor\!\rfloor \setminus X\lfloor\!\lfloor x \rfloor\!\rfloor \neq \emptyset$ since $u \notin X\lfloor\!\lfloor x \rfloor\!\rfloor$. For the rest of this proof, fix some arbitrary vertices $r \in X\lfloor\!\lfloor u \rfloor\!\rfloor \cap X\lfloor\!\lfloor x \rfloor\!\rfloor$ and $s \in X\lfloor\!\lfloor u \rfloor\!\rfloor \setminus X\lfloor\!\lfloor x \rfloor\!\rfloor$. We refer to Figure 6.8 for an illustration of the proof from here onwards.

By Definition 6.26, there are source paths P_r via r to u and P_s via s to u that intersect X only in r and s, respectively. Also, there is a source path P to x such that $P \cap X = \{r\}$ since $r \in X\lfloor\!\lfloor x \rfloor\!\rfloor$. Suppose without loss of generality that s is to the right of P. The paths P_s and P cannot intersect between s and u. To see this, observe that if P_s crosses P after s but before r, then by starting with P and switching to P_s at the intersection point we get a source path to u that is not blocked by X. And if the crossing is after r, we can start with P_s and then switch to P when the paths intersect, which implies that $s \in X\lfloor\!\lfloor x \rfloor\!\rfloor$ contrary to assumption. Thus u is located to the right of P as well.

Extend P_s by going north-west from u until hitting P, which must happen somewhere in between r and x, and then following P to x. Denote this extended

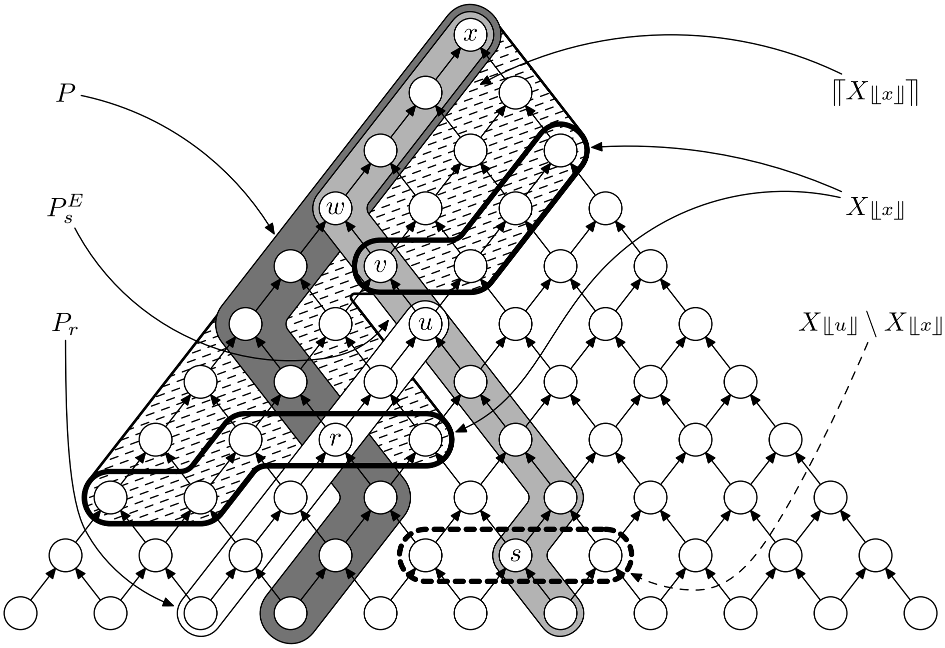

P
P_s^E
P_r
$\lceil\!\lceil X_{\llbracket x\rrbracket}\rceil\!\rceil$
$X_{\llbracket x\rrbracket}$
$X_{\llbracket u\rrbracket} \setminus X_{\llbracket x\rrbracket}$
x
w
v
u
r
s

Figure 6.8: Illustration of proof of Lemma 6.54 that all of X is needed to hide x.

path by P_s^E and let w be the vertex starting from which P_s^E and P coincide. The path P_s^E must intersect X in some more vertex after s since $s \notin X_{\llbracket x\rrbracket}$. Pick any $v \in P_s^E \cap (X \setminus \{s\})$. By construction, v must be located strictly between u and w. We claim that $X \setminus \{v\}$ hides v. This contradicts the tightness of X and the lemma follows.

To prove the claim, consider any source path P_v to v and assume that $P_v \cap (X \setminus \{v\}) = \emptyset$. Then, in particular, $r \notin P_v$. Suppose that P_v passes to the left of r. By planarity, P_v must intersect P somewhere above r. But if so, we can construct a source path P' to x that starts like P_v and switches to P at this intersection point. We get $P' \cap X = \emptyset$, which contradicts $x \in X_{\llbracket x\rrbracket}$. If instead P_v passes r on the right, then P_v must cross P_r in order to get to v. This implies that there is a source path P'' to u such that $P'' \cap X = \emptyset$, namely the path obtained by starting to go along P_v and then changing to P_r when the two paths intersect above r. Thus we get a contradiction in this case as well. Hence, $X \setminus \{v\}$ blocks any source path to v as claimed. □

The Ice-Cream Cone Lemma 6.48 now follows. Thereby, the proof of the lower bound on the black-white pebbling price of pyramid graphs in Theorem 6.24 on page 71 is complete.

Part III

On Space in Resolution

Chapter 7

A Simplified Way of Proving Trade-offs

As a warm-up before launching into the proofs of the results that are the main contribution of this thesis, in this chapter we present our simplification of the length-space trade-off result in [45], and show how the same ideas can be used to prove other related theorems. We also point out two key ingredients needed for our proofs to work and discuss possible conclusions to be drawn regarding proving trade-off results for resolution. This chapter is based on [61], to appear as one of the results in [63].

We will need the following easy observation.

Observation 7.1. *Suppose that $F = G \wedge H$ where G and H are unsatisfiable CNF formulas over disjoint sets of variables. Then any resolution refutation $\pi : F \vdash 0$ must contain a refutation of either G or H.*

Proof. By induction, we can never resolve a clause derived from G with a clause derived from H, since the sets of variables of the two clauses are disjoint. □

7.1 A Proof of Hertel and Pitassi's Trade-off Result

We show the following version of the length-variable space trade-off theorem of Hertel and Pitassi [45], with somewhat improved parameters and a very much simpler proof.

Theorem 3.6 (restated). *There is a family of CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that:*

- *The variable space of refuting F_n in resolution is $\mathit{VarSp}(F_n \vdash 0) = \Theta(n)$.*
- *Any refutation $\pi : F_n \vdash 0$ in minimal variable space has length $\exp(\Omega(\sqrt{n}))$.*
- *Adding at most 2 extra units of storage, one can obtain a refutation π' in space $\mathit{VarSp}(\pi') = \mathit{VarSp}(F_n \vdash 0) + 2 = \Theta(n)$ and length $L(\pi') = \mathrm{O}(n)$, i.e., linear in the formula size.*

We note that the CNF formulas used by Hertel and Pitassi, as well as those in our proof, have clauses of width $\Theta(n)$.

Proof of Theorem 3.6. Let G_n be CNF formulas as in Theorem 4.18 having size $\Theta(n)$, refutation length $L(G_n \vdash 0) = \exp(\Omega(n))$, and refutation clause space $Sp(G_n \vdash 0) = \Theta(n)$. Let us define $g(n) = \mathit{VarSp}(G_n \vdash 0)$ to be the refutation variable space of the formulas. Then it holds that $\Omega(n) = g(n) = \mathrm{O}\big(n^2\big)$.

Let H_m be the formulas

$$H_m = y_1 \wedge \cdots \wedge y_m \wedge (\overline{y}_1 \vee \cdots \vee \overline{y}_m) \ . \tag{7.1}$$

It is not hard to see that there are resolution refutations $\pi : H_m \vdash 0$ in length $L(\pi) = 2m + 1$ and variable space $\mathit{VarSp}(\pi) = 2m$, and that $L(H_m \vdash 0) = 2m + 1$ and $\mathit{VarSp}(H_m \vdash 0) = 2m$ are also the lower bounds (all clauses must be used in any refutation, and the minimum space refutation must start by downloading the wide clause and some unit clause, and then resolve).

Now define

$$F_n = G_n \wedge H_{\lfloor g(n)/2 \rfloor + 1} \tag{7.2}$$

where G_n and $H_{\lfloor g(n)/2 \rfloor + 1}$ have disjoint sets of variables. By Observation 7.1, any resolution refutation of F_n refutes either G_n or $H_{\lfloor g(n)/2 \rfloor + 1}$. We have

$$\mathit{VarSp}\big(H_{\lfloor g(n)/2 \rfloor + 1} \vdash 0\big) = 2 \cdot (\lfloor g(n)/2 \rfloor + 1) > g(n) = \mathit{VarSp}(G_n \vdash 0) \ , \tag{7.3}$$

so a resolution refutation in minimal variable space must refute G_n in length $\exp(\Omega(n))$. However, allowing at most two more literals in memory, the resolution refutation can disprove the formula $H_{\lfloor g(n)/2 \rfloor + 1}$ instead in length linear in the (total) formula size.

Thus, we have a formula family $\{F_n\}_{n=1}^{\infty}$ of size $\Omega(n) = S(F_n) = \mathrm{O}\big(n^2\big)$ refutable in length and variable space both linear in the formula size, but where any minimum variable space refutation must have length $\exp(\Omega(n))$. Adjusting the indices as needed, we get a formula family with a trade-off of the form stated in Theorem 3.6. □

7.2 Some Other Trade-off Results for Resolution

Using a similar trick as in the previous section, we can prove the following length-clause space trade-off.

Theorem 2.8 (restated). *There is a family of k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that:*

- *The minimal clause space of refuting F_n is $Sp(F_n \vdash 0) = \Theta\big(\sqrt[3]{n}\big)$.*
- *Any resolution refutation $\pi : F_n \vdash 0$ in minimal clause space must have length $L(\pi) = \exp\big(\Omega\big(\sqrt[3]{n}\big)\big)$.*

- *There are resolution refutations $\pi' : F_n \vdash 0$ in asymptotically minimal clause space $Sp(\pi') = \mathrm{O}\big(Sp(F_n \vdash 0)\big)$ and length $L(\pi') = \mathrm{O}(n)$, i.e., linear in the formula size.*

The same game can be played with refutation width as well.

Theorem 7.2. *There is a family of k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that:*

- *The minimal width of refuting F_n is $W(F_n \vdash 0) = \Theta\big(\sqrt[3]{n}\big)$.*
- *Any resolution refutation $\pi : F_n \vdash 0$ in minimal width must have length $L(\pi) = \exp\big(\Omega\big(\sqrt[3]{n}\big)\big)$.*
- *There are resolution refutations $\pi' : F_n \vdash 0$ in width $W(\pi') = \mathrm{O}\big(W(F_n \vdash 0)\big)$ and length $L(\pi') = \mathrm{O}(n)$.*

We only present the proof of Theorem 2.8, as Theorem 7.2 is proved in exactly the same manner.

Proof of Theorem 2.8. Let G_n be a 3-CNF formula family as in Theorem 4.18 having size $\Theta(n)$, refutation length $L(G_n \vdash 0) = \exp(\Theta(n))$, and refutation clause space $Sp(G_n \vdash 0) = \Theta(n)$. Let H_m be a 3-CNF formula family as in Theorem 4.21 of size $\Theta\big(m^3\big)$ such that $L(H_m \vdash 0) = \mathrm{O}\big(m^3\big)$ and $Sp(H_m \vdash 0) = \Theta(m)$. Define

$$g(n) = \min\big\{m \mid Sp(H_m \vdash 0) > Sp(G_n \vdash 0)\big\} \ . \tag{7.4}$$

Note that since $Sp(H_m \vdash 0) = \Omega(m)$ and $Sp(G_n \vdash 0) = \mathrm{O}(n)$, we know that $g(n) = \mathrm{O}(n)$.

Now as before let $F_n = G_n \wedge H_{g(n)}$, where G_n and $H_{g(n)}$ have disjoint sets of variables. By Observation 7.1, any resolution refutation of F_n is a refutation of either G_n or $H_{g(n)}$. Since $g(n)$ has been chosen so that $Sp\big(H_{g(n)} \vdash 0\big) > Sp(G_n \vdash 0)$, a refutation in minimal clause space has to refute G_n, which requires exponential length. However, since $g(n) = \mathrm{O}(n)$, Theorem 4.21 tells us that there are refutations of $H_{g(n)}$ in length $\mathrm{O}\big(n^3\big)$ and clause space $\mathrm{O}(n)$. □

7.3 Making the Main Trick Explicit

The proofs of the theorems in Sections 7.1 and 7.2 come very easily; in fact almost *too* easily. What is it that makes this possible? In this and the next section, we want to highlight two key ingredients in the constructions.

The common paradigm for the proofs of Theorems 2.8, 3.6, and 7.2 is as follows. We are given two complexity measures M_1 and M_2 that we want to trade off against one another. We do this by finding formulas G_n and H_m such that

- The formulas G_n are very hard with respect to the first resource measured by M_1, while $M_2\big(G_n\big)$ is at most some (more or less trivial) upper bound.

- The formulas H_m are very easy with respect to M_1, but there is some non-trivial *lower* bound on the usage $M_2\big(H_m\big)$ of the second resource.
- The index $m = m(n)$ is chosen so as to minimize $M_2\big(H_{m(n)}\big) - M_2\big(G_n\big) > 0$, i.e., so that $H_{m(n)}$ requires *just a little bit* more of the second resource than G_n.

Then for $F_n = G_n \land H_{m(n)}$, if we demand that a resolution refutation π must use the minimal amount of the second resource, it will have to use a large amount of the first resource. However, relaxing the requirement on the second resource by the very small expression $M_2\big(H_{m(n)}\big) - M_2\big(G_n\big)$, we can get a refutation π' using small amounts of both resources.

Clearly, the formula families $\{F_n\}_{n=1}^{\infty}$ that we get in this way are "redundant" in the sense that each formula F_n is the conjunction of two formulas G_n and H_m which are themselves already unsatisfiable. Formally, we say that a formula F is *minimally unsatisfiable* if F is unsatisfiable, but removing any clause $C \in F$ makes the remaining subformula $F \setminus \{C\}$ satisfiable. We note that if we would add the requirement in Sections 7.1 and 7.2 that the formulas under consideration should be minimally unsatisfiable, the proof idea outlined above fails completely. In contrast, the result in [45] seems to be independent of any such conditions. What conclusions can be drawn from this?

On the one hand, trade-off results for minimally unsatisfiable formulas seem more interesting, since they tell us something about a property that some natural formula family has, rather than about some funny phenomena arising because we glue together two totally unrelated formulas.

On the other hand, one could argue that the main motivation for studying space is the connection to memory requirements for proof search algorithms, for instance algorithms using clause learning. And for such algorithms, a minimality condition might appear somewhat arbitrary. There are no guarantees that "real-life" formulas will be minimally unsatisfiable, and most probably there is no efficient way of testing this condition.[1] So in practice, trade-off results for non-minimal formulas might be just as interesting.

7.4 An Auxiliary Trick for Variable Space

A second important reason why our proof of Theorem 3.6 gives sharp results is that we are allowed to use CNF formulas of growing width. It is precisely because of this that we can easily construct the needed formulas H_m that are hard with respect to variable space but easy with respect to length. If we would have to restrict ourselves to k-CNF formulas for k constant, it would be much more difficult to

[1] The problem of deciding minimal unsatisfiability is NP-hard but not known to be in NP. Formally, a language L is in the complexity class DP if and only if there are two languages $L_1 \in \mathsf{NP}$ and $L_2 \in \mathsf{co\text{-}NP}$ such that $L = L_1 \cap L_2$ [64]. MINIMAL UNSATISFIABILITY is DP-complete [65], and it seems to be commonly believed that $\mathsf{DP} \not\subseteq \mathsf{NP} \cup \mathsf{co\text{-}NP}$.

find such examples. Although the formulas in Theorem 4.21 could be plugged in to give a slightly weaker trade-off, we are not aware of any family of k-CNF formulas that can provably give the very sharp result in Theorem 3.6. (Note, though, that the formula families used in the proofs of Theorems 2.8 and 7.2 consist of k-CNF formulas).

This is not the only example of a space measure behaving badly for formulas of growing width. We have already discussed the lower bound $Sp(F \vdash 0) \geq W(F \vdash 0) - W(F) + 3$ on clause space in terms of length in Theorem 4.22, and the result, proven in this thesis, that the inequality above is asymptotically strict in the sense that there are k-CNF formula families F_n with $W(F_n \vdash 0) = \mathrm{O}(1)$ but $Sp(F_n \vdash 0) = \omega(1)$.

However, if we are allowed to consider formulas of growing width, the fact that the inequality in Theorem 4.22 is not tight is entirely trivial. Namely, let us say that a CNF formula F is *k-wide* if all clauses in F have size at least k. In [39], it was proven that for F a k-wide unsatisfiable CNF formula it holds that $Sp(F \vdash 0) \geq k + 2$. So in order to get a formula family F_n such that $W(F_n \vdash 0) - W(F_n) = \mathrm{O}(1)$ but $Sp(F_n \vdash 0) = \omega(1)$, just pick some suitable formulas $\{F_n\}_{n=1}^{\infty}$ of growing width.

In our opinion, these phenomena are clearly artificial. Since every CNF formula can be rewritten as an equivalent k-CNF formula without increasing the size more than linearly (using extension variables), the right approach when studying space measures in resolution seems to be to require that the formulas under study should have constant width.

As a final comment before moving on to our main results, we note that the open trade-off questions mentioned in Section 12.3 do not suffer from the technical problems discussed above.

Chapter 8

A Separation of Space and Width

In this chapter we prove Theorem 2.1 and Corollary 2.2, thus providing a first (weak) separation of clause space and width. We do this by establishing asymptotically tight bounds on the clause space of refuting pebbling contradictions over binary trees. This result has been published as [60], the full-length version of which is to appear as [62].

We note that the result in this chapter is subsumed by what will be presented in Chapter 9 in the sense that the results there hold for more general classes of graphs and formulas. However, the actual constants that we get for pebbling contradictions over binary trees are much better in this chapter, and in fact could be pushed to very nearly optimal if one wanted to (although we opt for a simpler proof with slightly worse constants in what follows).

8.1 Overview of Space-Width Separation Proof

Let us start by describing how we implement the general approach in Section 3.3 in this concrete case. Recall that our guiding intuition is that some kind of pebbling–resolution-correspondence as sketched in Section 3.2 should hold, but since we are not able to interpret clauses in terms of pebbles in such a way that resolution derivation steps comply with the rather restrictive set of rules of the black-white pebble game, we instead try to modify the pebbling rules.

The most important change in the new pebble game that we construct, compared to the black-white pebble game in Definition 5.1, is that we allow "sliding moves" of black pebbles downwards and of white pebbles upwards in the graph. As we will see, this leads to serious technical difficulties. Another, less far-reaching, modification of the game can be motivated as follows. Looking at the clauses and pebbles in Figure 3.3 on page 29, it somehow seems that the white pebbles on s and t are relevant only for the black pebble on v. The black pebble on u corresponding to $\bigvee_{i=1}^{d} x(u)_i$ is wholly independent of these white pebbles, although strictly from a pebbling perspective it is not, since the white pebble on s is below u. It turns out that it is important to formalize and keep track of this "dependence" relation between black and white pebbles, so we will have to label each black pebble in the graph with the white-pebbled vertices it depends on. Because of this property, we name the new game the *labelled pebble game* (Definition 8.5).

Once we have defined this labelled pebble game in Section 8.2, we continue according to the proof outline in Section 3.3. However, for reasons that will be clear below, we restrict our attention to binary trees.

In Section 8.3, we show that a resolution refutation of a pebbling contradiction defined over a binary tree induces a pebbling of this tree in our modified pebble game.

Theorem 8.1. *Let $Peb^d_{T_h}$ denote the pebbling contradiction of degree $d \geq 1$ over the complete binary tree T_h of height h. Then there is a translation function from sets of clauses derived from $Peb^d_{T_h}$ into sets of pebbles in T_h such that any resolution refutation π of $Peb^d_{T_h}$ corresponds to a labelled pebbling $\mathcal{L}_\pi$ of T_h under this translation.*

In Section 8.4, we prove that if the number of variables d associated to each vertex is at least 2, then the cost of the labelled pebbling $\mathcal{L}_\pi$ in Theorem 8.1 is related to the space of the resolution refutation π.

Theorem 8.2. *If π is a resolution refutation of a pebbling contradiction $Peb^d_{T_h}$ of degree $d > 1$, then the cost of the associated labelled pebbling $\mathcal{L}_\pi$ is asymptotically bounded by the space of π, or in formal notation $\mathsf{cost}(\mathcal{L}_\pi) = \mathrm{O}(Sp(\pi))$.*

Finally, we need a lower bound for the pebbling price of binary trees in the labelled pebble game.

Theorem 8.3. *Any complete labelled pebbling $\mathcal{L}$ of T_h must have cost at least linear in the tree height h. That is, the labelled pebbling price of T_h is $\mathsf{L\text{-}Peb}(T_h) = \Omega(h)$.*

We establish this result by transforming labelled pebblings to pebblings in the standard black-white pebble game and then using known bounds on the black-white pebbling price of binary trees (Theorem 5.2). The technically quite complicated proof is given in Section 8.5. The reason that we consider only binary trees is that the analogue of Theorem 8.3 does not hold for more general DAGs. For instance, it is false for the pyramid graph in Figure 3.3 (as is shown in Lemma 8.6).

Putting all of this together, we can now prove the main results in this chapter.

Theorem 2.1 (restated). *Let T_h denote the complete binary tree of height h and $Peb^d_{T_h}$ the pebbling contradiction of degree $d > 1$ defined over T_h. Then the space of refuting $Peb^d_{T_h}$ by resolution is $Sp(Peb^d_{T_h} \vdash 0) = \Theta(h)$.*

Proof. The upper bound $Sp(Peb^d_{T_h} \vdash 0) = \mathrm{O}(h)$ is the easy part. It follows from Theorem 5.2 and Proposition 5.10, since the refutation space of a pebbling contradiction is upper-bounded by the black pebbling price of its underlying graph, and binary trees of height h have black pebbling price $\mathrm{O}(h)$.

For the lower bound, let π be any resolution refutation of $Peb^d_{T_h}$. Consider the associated labelled pebbling $\mathcal{L}_\pi$ provided by Theorem 8.1. On the one hand, we know that $\mathsf{cost}(\mathcal{L}_\pi) = \mathrm{O}(Sp(\pi))$ by Theorem 8.2, provided that $d > 1$. On the other hand, Theorem 8.3 tells us that the cost of any pebbling of T_h is $\Omega(h)$, so in particular we must have $\mathsf{cost}(\mathcal{L}_\pi) = \Omega(h)$. Combining these two bounds on $\mathsf{cost}(\mathcal{L}_\pi)$, we see that $Sp(\pi) = \Omega(h)$. Since this bound holds for any resolution refutation π, it follows that the minimum clause space of refuting $Peb^d_{T_h}$ is $Sp(Peb^d_{T_h} \vdash 0) = \Omega(h)$ for $d > 1$. The theorem follows. □

We know that the pebbling contradiction Peb^d_G is a $(2{+}d)$-CNF formula, and for fixed d the size of the formula is linear in the number of vertices of G. Thus, for binary trees, $Peb^d_{T_h}$ has size exponential in the tree height h. Also, Proposition 5.7 tells us that Peb^d_G can be refuted in width $W(Peb^d_G \vdash 0) = \mathrm{O}(d)$ for any graph G.

The separation of space and width in Corollary 2.2 follows from this if we fix $d > 1$ and set $F_n = Peb^d_{T_h}$ for $h = \lfloor \log(n+1) \rfloor$ in Theorem 2.1.

Corollary 2.2 (restated). *For all $k \geq 4$, there is a family of k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\mathrm{O}(n)$ such that $W(F_n \vdash 0) = \mathrm{O}(1)$ but $Sp(F_n \vdash 0) = \Theta(\log n)$.*

8.2 The Labelled Pebble Game

In this section, we present our modified pebble game used for analyzing resolution derivations. We then argue that for binary trees, we get essentially the same bound on pebbling price in this new pebble game as in the black-white pebble game of Definition 5.1.

Our first modification of the pebble game is to change the rule for white pebble removal so that a white pebble can be removed from a vertex when a black pebble is placed on that same vertex. This will make the correspondence between pebblings and resolution derivations much more natural. Clearly, this is only a minor adjustment, and it is easy to prove formally that it does not really change anything.

Our second, and far more substantial, modification of the pebble game is motivated by the fact that in general, a resolution refutation has no obvious reason to follow our intuition that it should somehow correspond to a pebbling of the underlying graph. Since pebbles are induced by clauses, if at some derivation step the refutation chooses to erase “the wrong clause” from the point of view of the induced pebble configuration, this can lead to pebbles just disappearing. This is all in order for black pebbles, but if we allow uncontrolled removal of white pebbles, we cannot hope for any nontrivial lower bounds on pebbling price (just white-pebble the two predecessors of the target, then black-pebble the target itself, and finally remove the white pebbles).

Our solution to this problem is to keep track of exactly which white pebbles have been used to get a black pebble on a vertex. Loosely put, removing a white pebble from a vertex v without placing a black pebble on the same vertex should be in order, provided that all black pebbles placed on vertices above v in the DAG with the help of the white pebble on v are removed as well. We define a pebble *subconfiguration* to consist of a black pebble together with all the white pebbles this black pebble depends on, and require that if a white pebble in a subconfiguration is removed, then all other pebbles in this subconfiguration must be removed as well.

Another problem is that some resolution derivation steps can lead to what looks like "backward" pebbling moves, with white pebbles moving upwards and black pebbles downwards in the DAG. This problem turns out to be even more serious. We try to get around it by introducing an order relation on pebble subconfigurations, where the intuition is that "stronger" pebble subconfigurations are "closer" to the final goal of getting the target black-pebbled. Using this order relation, the backward pebbling moves can be characterized as moves from stronger to weaker pebble subconfigurations, so we add a pebbling rule allowing such moves.

To define this modified pebble game formally, we need some notation and terminology. We write $p, q, r, s, u, v, w, x, y$ to denote arbitrary vertices of the DAG G and U, V, W to denote arbitrary subsets of vertices. We will always use z to denote the unique target vertex of the DAG. Also, recall that $\mathit{succ}(v)$ denotes the immediate successor of v and $\mathit{pred}(v)$ denotes the immediate predecessors. For a leaf v we have $\mathit{pred}(v) = \emptyset$, and for the target z we have $\mathit{succ}(z) = \emptyset$. We say that w is *below* v if there is a path from w to v and *above* v if there is a path from v to w. If in addition $v \neq w$, the vertex w is said to be *strictly* below/above v. We say that v and w are *unrelated* if v is neither above nor below w. The vertex set W is *(strictly) below* v if all $w \in W$ are (strictly) below v.

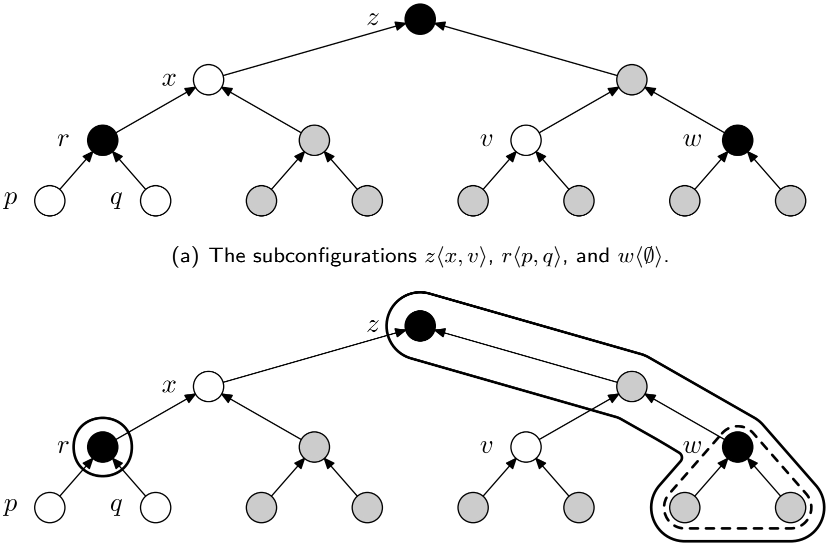
We now present the concept used to "label" each black pebble with the set of white pebbles (if any) that this black pebble is dependent on. The intuition behind the next definition is that $v\langle W\rangle$ should denote a black pebble on v together with the white pebbles W below v with the help of which we have been able to place the black pebble on v.

Definition 8.4 (Pebble subconfiguration). For a vertex v and a set of vertices W strictly below v, we say that $v\langle W\rangle$ is a *pebble subconfiguration* with a black pebble on v supported by white pebbles on $w \in W$. The black pebble on v in $v\langle W\rangle$ is said to be *dependent* on the white pebbles in W. We refer to $v\langle\emptyset\rangle$ as an *independent black pebble.*

The *cover* of $v\langle W\rangle$, denoted $\mathit{cover}(v\langle W\rangle)$, consists of all vertices U such that there is a path $P : u \rightsquigarrow v$ from $u \in U$ to v that does not intersect W, i.e., $P \cap W = \emptyset$. If $\mathit{cover}(v_1\langle W_1\rangle) \subseteq \mathit{cover}(v_2\langle W_2\rangle)$, we say that $v_1\langle W_1\rangle$ is *covered by* $v_2\langle W_2\rangle$ and write $v_1\langle W_1\rangle \preceq v_2\langle W_2\rangle$. If $\mathit{cover}(v_1\langle W_1\rangle) \subsetneqq \mathit{cover}(v_2\langle W_2\rangle)$, we write $v_1\langle W_1\rangle \prec v_2\langle W_2\rangle$.

We use $\mathbb{L}$ to denote a set of pebble subconfigurations and refer to such a set as a *labelled pebble configuration* or an *L-configuration.* The cover of an L-configuration $\mathbb{L}$ is defined as $\mathit{cover}(\mathbb{L}) = \bigcup_{v\langle W\rangle \in \mathbb{L}} \mathit{cover}(v\langle W\rangle)$, and we write $\mathbb{L}_1 \preceq \mathbb{L}_2$ if $\mathit{cover}(\mathbb{L}_1) \subseteq \mathit{cover}(\mathbb{L}_2)$.

In the following, when we specify the set W of white-pebbled vertices in $v\langle W\rangle$



(a) The subconfigurations $z\langle x, v\rangle$, $r\langle p, q\rangle$, and $w\langle\emptyset\rangle$.

(b) The covers $cover(z\langle x, v\rangle)$, $cover(r\langle p, q\rangle)$, and $cover(w\langle\emptyset\rangle)$ (dashed).

Figure 8.1: Three pebble subconfigurations and their covered vertices.

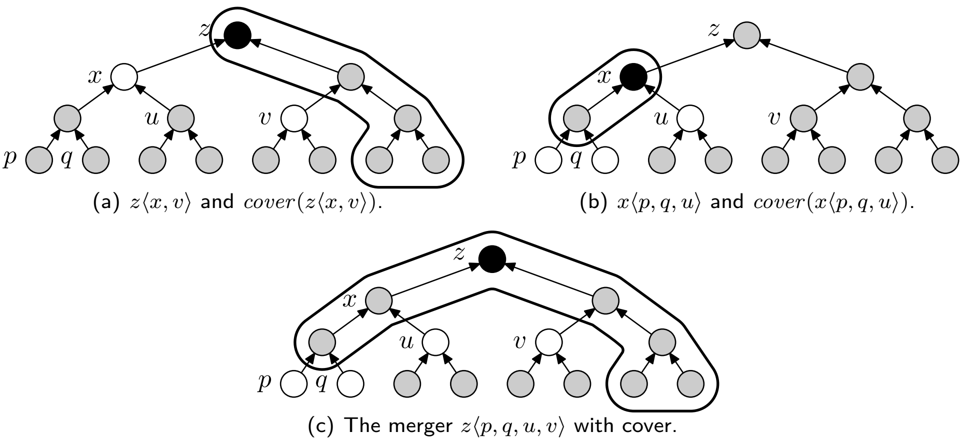
by enumerating the members of W, we will abuse notation somewhat by omitting the curly brackets inside $\langle$ and $\rangle$ around this set.

For an illustration of Definition 8.4, see Figure 8.1. Note that $w\langle\emptyset\rangle \prec z\langle x, v\rangle$ since $cover(w\langle\emptyset\rangle) \subsetneqq cover(z\langle x, v\rangle)$ (see Figure 8.1(b)). We remark that $\preceq$ is an order relation on pebble subconfigurations, as the notation suggests, and that the minimal elements are subconfigurations $v\langle pred(v)\rangle$.

Our modified pebble game is defined in terms of moves not of individual pebbles, but of entire pebble subconfigurations. In this pebble game, a black pebble on v is always placed together with white pebbles on $pred(v)$ below (except for at the leaves where $pred(v) = \emptyset$). Removals of white pebbles are always allowed, but since we can remove only a whole subconfiguration, the removal rule ensures that any black pebble dependent on the removed white pebbles is removed as well. A "traditional" removal of a white pebble from w corresponds to merging two subconfigurations $v\langle V\rangle$ and $w\langle W\rangle$ into $v\langle (V \cup W) \setminus \{w\}\rangle$ and then erasing $v\langle V\rangle$ and $w\langle W\rangle$ (see Figure 8.2 for an example). Finally, we allow *reversal* moves to weaker subconfigurations. The formal definition is as follows.

Definition 8.5 (Labelled pebble game). For G any DAG with unique target z, a *labelled pebbling*, or *L-pebbling*, on G is a sequence $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$ of labelled pebble configurations such that for all t it holds that $\mathbb{L}_t \neq \mathbb{L}_{t+1}$ and $\mathbb{L}_{t+1}$ is obtained from $\mathbb{L}_t$ by one of the following rules:

Introduction $\mathbb{L}_{t+1} = \mathbb{L}_t \cup \{v\langle pred(v)\rangle\}$.



(a) $z\langle x, v\rangle$ and $cover(z\langle x, v\rangle)$.

(b) $x\langle p, q, u\rangle$ and $cover(x\langle p, q, u\rangle)$.

(c) The merger $z\langle p, q, u, v\rangle$ with cover.

Figure 8.2: Two pebble subconfigurations and their merger.

Erasure $\mathbb{L}_{t+1} = \mathbb{L}_t \setminus \{v\langle V\rangle\}$ for $v\langle V\rangle \in \mathbb{L}_t$.

Merger $\mathbb{L}_{t+1} = \mathbb{L}_t \cup \big\{v\langle (V \cup W) \setminus \{w\}\rangle\big\}$ for $v\langle V\rangle, w\langle W\rangle \in \mathbb{L}_t$ with $w \in V$. We denote this subconfiguration $\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$, where the pair of subconfigurations $v\langle V\rangle$, $w\langle W\rangle$ is always ordered so that $w \in V$, and refer to it as a *merger on* w.

Reversal $\mathbb{L}_{t+1} = \mathbb{L}_t \cup \{v\langle V\rangle\}$ if $v\langle V\rangle \preceq u\langle U\rangle$ for some $u\langle U\rangle \in \mathbb{L}_t$.

Let $Bl(\mathbb{L}_t) = \bigcup\{v \mid v\langle W\rangle \in \mathbb{L}_t\}$ denote the set of all black pebbles in $\mathbb{L}_t$ and $Wh(\mathbb{L}_t) = \bigcup\{W \mid v\langle W\rangle \in \mathbb{L}_t\}$ the set of all white pebbles. Then the *cost* of an L-configuration $\mathbb{L}$ is $\mathit{cost}(\mathbb{L}) = |Bl(\mathbb{L}) \cup Wh(\mathbb{L})|$, and the cost of an L-pebbling $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$ is $\max_{t\in[\tau]}\{\mathit{cost}(\mathbb{L}_t)\}$.

A *complete labelled pebbling* of G is an L-pebbling $\mathcal{L}$ such that $\mathbb{L}_0 = \emptyset$ and $\mathbb{L}_\tau = \{z\langle\emptyset\rangle\}$. The *labelled pebbling price* of G, denoted $\mathsf{L\text{-}Peb}(G)$, is the minimum cost of any complete L-pebbling of G.

The "backward" pebbling moves mentioned at the beginning of this section are moves according to the reversal rule. It can be shown that the L-pebble game without reversal moves is essentially just a disguised version of the ordinary black-white pebble game. Arguing very informally, it seems plausible that making reversals in an L-pebbling should only "weaken" the pebble configurations (for example, reversing from $z\langle x, v\rangle$ to $w\langle\emptyset\rangle$ in Figure 8.1), and that it should therefore be possible to eliminate all reversal moves from a pebbling without affecting the pebbling cost.

Unfortunately, this intuition does not hold in general.

Lemma 8.6. *There are families of DAGs $\{G_n\}_{n=1}^{\infty}$ such that $\mathsf{BW\text{-}Peb}(G_n)$ goes to infinity with n but $\mathsf{L\text{-}Peb}(G_n)$ is constant.*

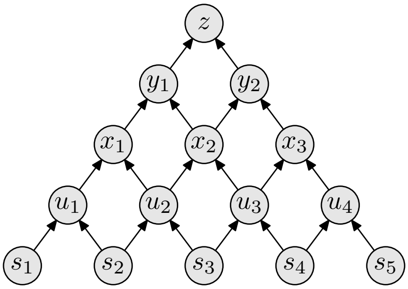


Figure 8.3: Example pyramid in proof of Lemma 8.6 that L-pebbling price collapses.

Proof. Consider the pyramid graphs Π_h (with Π_4 shown in Figure 8.3). We know from Theorem 5.3 that $\mathsf{BW\text{-}Peb}(\Pi_h) = h/2 + \mathrm{O}(1)$. We prove by induction that Π_h can be L-pebbled with 4 pebbles if we allow reversal moves of black pebbles downwards.

The base case for a pyramid of height 1 is clear. For the induction step, suppose that we have been able to get to the pebble subconfiguration $y_2\langle\emptyset\rangle$ in Figure 8.3 in L-pebbling cost at most 4. We show how to get a black pebble on the target z by an introduction move and then move the white pebbles downwards one level at a time until we reach the sources.

Introducing $z\langle y_1, y_2\rangle$ and merging $z\langle y_1, y_2\rangle$ with $y_2\langle\emptyset\rangle$ on y_2, we get $z\langle y_1\rangle$. Next, reverse $y_2\langle\emptyset\rangle$ to $x_2\langle\emptyset\rangle$ (this is a legal reversal move, since $\mathit{cover}(x_2\langle\emptyset\rangle) \subseteq \mathit{cover}(y_2\langle\emptyset\rangle)$). Conclude this first subsequence of L-pebbling moves by erasing $z\langle y_1, y_2\rangle$ and $y_2\langle\emptyset\rangle$.

Now we have the L-configuration $\bigl\{z\langle y_1\rangle, x_2\langle\emptyset\rangle\bigr\}$. Introduce $y_1\langle x_1, x_2\rangle$, merge $z\langle y_1\rangle$ and $y_1\langle x_1, x_2\rangle$ on y_1 resulting in $z\langle x_1, x_2\rangle$, and erase $z\langle y_1\rangle$ and $y_1\langle x_1, x_2\rangle$. Then merge $z\langle x_1, x_2\rangle$ and $x_2\langle\emptyset\rangle$ on x_2 to get $z\langle x_1\rangle$. As above, conclude the subsequence of moves by reversing $x_2\langle\emptyset\rangle$ to $u_2\langle\emptyset\rangle$ and then erasing $z\langle x_1, x_2\rangle$ and $x_2\langle\emptyset\rangle$.

The next round of moves is entirely analogous: Introduce $x_1\langle u_1, u_2\rangle$, merge with $z\langle x_1\rangle$ to get $z\langle u_1, u_2\rangle$, and then erase the merged subconfigurations. Then merge $z\langle u_1, u_2\rangle$ with $u_2\langle\emptyset\rangle$ resulting in $z\langle u_1\rangle$, reverse $u_2\langle\emptyset\rangle$ to $s_2\langle\emptyset\rangle$, and erase $z\langle u_1, u_2\rangle$ and $u_2\langle\emptyset\rangle$.

At the start of the final subsequence of moves, we have the L-configuration $\bigl\{z\langle u_1\rangle, s_2\langle\emptyset\rangle\bigr\}$. Introducing $u_1\langle s_1, s_2\rangle$ and merging this subconfiguration with $z\langle u_1\rangle$ on u_1 results in $z\langle s_1, s_2\rangle$. Introducing $s_1\langle\emptyset\rangle$ and merging $z\langle s_1, s_2\rangle$ with $s_1\langle\emptyset\rangle$ and then $s_2\langle\emptyset\rangle$, we get $z\langle\emptyset\rangle$.

The cost of this pebbling is 4, and it is easy to see that it generalizes to pyramids of arbitrary height. □

For binary trees, however, we can prove that the L-pebbling price and the black-white pebbling price coincide asymptotically.

Theorem 8.7. *For a complete binary tree T, $\mathsf{L\text{-}Peb}(T) = \Theta\bigl(\mathsf{BW\text{-}Peb}(T)\bigr)$.*

The technically quite complicated proof of this fact, which is a cornerstone of our result, is presented in the final section of this chapter.

Given Theorem 8.7, the lower bound on L-pebbling price in Theorem 8.3 on page 98 follows.

Theorem 8.3 (restated). *L-Peb*$(T_h) = \Omega(h)$.

Proof. Theorem 8.7 says that *L-Peb*$(T_h) = \Theta\bigl($*BW-Peb*$(T_h)\bigr)$, and Theorem 5.2 on page 52 says that *BW-Peb*$(T_h) = \Theta(h)$. □

8.3 Resolution Derivations Induce Labelled Pebblings

The next step in our proof is to show that sets of clauses can be interpreted in terms of pebble configurations in such a way that resolution derivations induce legal labelled pebblings.

For simplicity, from now on let us write $v_1, \ldots, v_d$ instead of $x(v)_1, \ldots, x(v)_d$ for the d variables associated with the vertex v in a dth degree pebbling contradiction.

Definition 8.8. Assume that G is a DAG with a unique target z and all vertices having indegree 0 or 2. Then we define $^*Peb^d_G = Peb^d_G \setminus \{\overline{z}_1, \ldots, \overline{z}_d\}$ to be the pebbling contradiction with target axioms removed. If $pred(r) = \{p, q\}$, the *axioms for* r is the set $Ax^d(r) = \bigl\{\overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} r_l \mid i, j \in [d]\bigr\}$, and for r a source we let $Ax^d(r) = \bigl\{\bigvee_{i=1}^{d} r_i\bigr\}$. For a set of vertices V, we define $Ax^d(V) = \bigl\{Ax^d(v) \mid v \in V\bigr\}$.

Let us first observe that instead of refutations of Peb^d_G, we can just as well study derivations of $\bigvee_{i=1}^{d} z_i$ from $^*Peb^d_G$. This will help us to avoid some artificial technicalities when defining the correspondence between resolution derivations and L-pebblings.

Lemma 8.9. *For any DAG G with a unique target z and all vertices having indegree 0 or 2, it holds that $Sp(Peb^d_G \vdash 0) = Sp(^*Peb^d_G \vdash \bigvee_{l=1}^{d} z_l)$. In particular, for every resolution refutation $\pi : Peb^d_G \vdash 0$ we can find a resolution derivation $\pi^* : {}^*Peb^d_G \vdash \bigvee_{l=1}^{d} z_l$ in the same space.*

Proof. For any resolution derivation $\pi^* : {}^*Peb^d_G \vdash \bigvee_{l=1}^{d} z_l$, we can get a resolution refutation of Peb^d_G from π^* in the same space by resolving $\bigvee_{l=1}^{d} z_l$ with all $\overline{z}_l$, $l = 1, \ldots, d$, in space 3.

In the other direction, for $\pi : Peb^d_G \vdash 0$ we can extract a derivation of $\bigvee_{l=1}^{d} z_l$ in at most the same space by simply omitting all downloads of and resolution steps on $\overline{z}_l$ in π, leaving the literals z_l in the clauses. Instead of the final empty clause 0 we get some clause $D \subseteq \bigvee_{l=1}^{d} z_l$, and since $^*Peb^d_G \nvDash D \subsetneqq \bigvee_{l=1}^{d} z_l$ and resolution is sound, we have $D = \bigvee_{l=1}^{d} z_l$. □

Now we try to develop some intuition for how clause configurations in a resolution derivation of $\bigvee_{i=1}^{d} z_i$ from $^*Peb^d_G$ should be translated into pebble configurations in the L-pebble game. Since we know from Lemma 8.6 that we cannot hope to

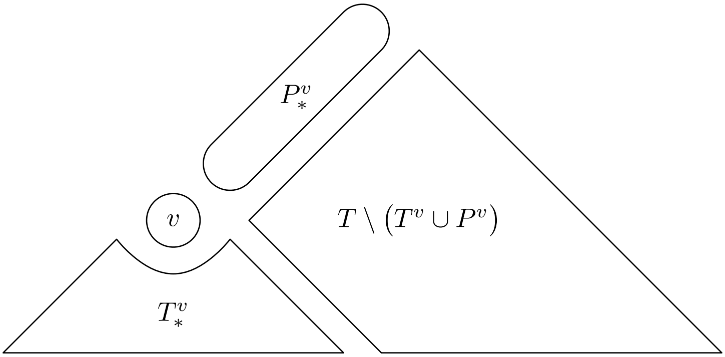


Figure 8.4: Referencing sets of vertices of a tree relative to a vertex.

get lower bounds for refutation space of pebbling contradictions over general DAGs by using the L-pebble game, from now on we concentrate exclusively on binary trees. To do this, we need some more notation and terminology. (Note that the notation below is somewhat different from that in Chapter 6.)

Definition 8.10. For a vertex v in a binary tree T, we let T^v denote the vertices in the complete binary subtree of T rooted at v, and $T^v_* = T^v \setminus \{v\}$ the vertices in T^v without its root v. We let P^v denote the vertices in the unique path from v to the root z of T and $P^v_* = P^v \setminus \{v\}$ the path without v.

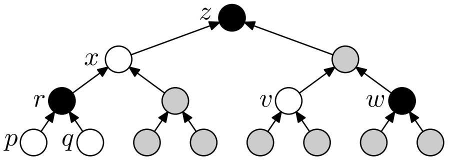
Definition 8.10 is illustrated in Figure 8.4. We blur the distinction somewhat between a tree T and the vertices in $V(T)$ and write, for instance, $T \setminus \big(T^v \cup P^v\big)$ instead of $V(T) \setminus \big(T^v \cup P^v\big)$ to denote all vertices in the tree unrelated to v.

In the standard black-white pebble game, if at some time t there is an independent black pebble on v, a pebbling need not place any pebbles on T^v after time t. As an analogy, if $\mathbb{C}_t \vDash \bigvee_{i=1}^{d} v_i$, it is not difficult to see that no axioms from $Ax^d(T^v)$ need be used in the resolution derivation after time t to derive $\bigvee_{i=1}^{d} z_i$. Therefore, it seems natural to think of a black pebble on v as derived truth $\bigvee_{i=1}^{d} v_i$ of v, and we want $\mathbb{C}_t$ to induce a subconfiguration $v\langle\emptyset\rangle$ if $\mathbb{C}_t \vDash \bigvee_{i=1}^{d} v_i$.

What kind of clause configuration should correspond to a dependent black pebble on v supported by white pebbles on W, i.e., a subconfiguration $v\langle W\rangle$? Well, one way of looking at $v\langle W\rangle$ is that this is the subconfiguration such that we would obtain an independent black pebble on v from it if the white pebbles on W were removed. But getting white pebbles *off* vertices is exactly as hard as getting black pebbles *on* vertices (compare with Proposition 6.8). In view of this, we can describe $v\langle W\rangle$ as the subconfiguration from which we can immediately derive $v\langle\emptyset\rangle$ by assuming black pebbles on W. And as to black pebbles, we just argued that they should correspond to clauses $\bigvee_{i=1}^{d} v_i$. Our conclusion is that $\mathbb{C}_t$ should induce $v\langle W\rangle$ if this clause configuration *together with assumed independent black pebbles on all* $w \in W$ implies an

$$\left[\begin{array}{l|c}\overline{x}_i \vee \overline{v}_j \vee \bigvee_{l=1}^{d} z_l & \\ \overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} r_l & i,j \in [d] \\ \bigvee_{l=1}^{d} w_l & \end{array}\right]$$

(a) Clause configuration $\mathbb{C}$.



(b) $\mathbb{L}(\mathbb{C}) = \{z\langle x, v\rangle,\ r\langle p, q\rangle,\ w\langle\emptyset\rangle\}$.

Figure 8.5: Example clause configuration $\mathbb{C}$ and its induced L-configuration $\mathbb{L}(\mathbb{C})$.

independent black pebble on v, i.e., if $\mathbb{C}_t \cup \left\{\bigvee_{i=1}^{d} w_i \mid w \in W\right\} \vDash \bigvee_{i=1}^{d} v_i$. Continuing our example from Figure 8.1, in Figure 8.5 we present a clause configuration corresponding to the given set of pebbles according to this intuitive understanding of induced pebble configurations.

Our formal definitions follow this intuition fairly closely, but since resolution derivations have no reason to be as well-behaved as to fit the description above, we need to add a number of technical details.

For white pebbles, it will simplify matters if we can ensure that they have the following property.

Definition 8.11. For a vertex v and a vertex set W strictly below v, if for every $w \in W$ there is a path $P : w \rightsquigarrow v$ not intersecting $W \setminus \{w\}$, we say that W is a *simple set below* v and that $v\langle W\rangle$ is a *simple subconfiguration*. $\mathbb{L}$ is a *simple L-configuration* if all subconfigurations $v\langle W\rangle \in \mathbb{L}$ are simple.

In the following, $\mathbb{B}(V)$ can be thought of as "truth of all vertices in V" and $All^{+}(V)$ as "truth of some vertex in V." We will be particularly interested in clauses $All^{+}(P^{v})$, i.e., clauses stating that some variable on the path from v to the root z is true.

Definition 8.12. Let $\mathbb{B}(V) = \left\{\bigvee_{i=1}^{d} v_i \mid v \in V\right\}$ and $All^{+}(V) = \bigvee_{v \in V} \bigvee_{i=1}^{d} v_i$.

Given a set of clauses $\mathbb{C}$ and a vertex v, if a vertex set $V \subseteq T \setminus P^{v}$ is such that $\mathbb{C} \cup \mathbb{B}(V) \vDash All^{+}(P^{v})$, we say that V is a *support* for v with respect to $\mathbb{C}$. If there is no $V' \subsetneqq V$ such that $\mathbb{C} \cup \mathbb{B}(V') \vDash All^{+}(P^{v})$ the support is *minimal*. If V is a support for v with respect to $\mathbb{C}$ such that $\mathbb{C} \cup \mathbb{B}(V) \nvDash All^{+}(P^{v}_{*}) = All^{+}(P^{v}) \setminus \bigvee_{i=1}^{d} v_i$, we say that v is *maximal* with respect to $\mathbb{C}$ and V.

We define the *supporting white pebbles* in the set V of the vertex v as $swp(v, V) = \left\{w \in V \cap T^{v}_{*} \mid P^{w}_{*} \cap V = \emptyset\right\}$.

When it is clear from context, we sometimes omit which support or vertex is minimal or maximal with respect to what. Note that $swp(v, V)$ is a simple set below v in the sense of Definition 8.11.

Definition 8.13 (Induced L-configuration). For a set of clauses $\mathbb{C}$ derived from $*Peb^{d}_{T}$, the *induced L-configuration* $\mathbb{L}(\mathbb{C})$ consists of all subconfigurations $v\langle V\rangle$ such that:

1. there is a minimal support $V' \subseteq T \setminus P^v$ for v with respect to $\mathbb{C}$,
2. v is maximal with respect to $\mathbb{C}$ and V', and
3. $V = \mathit{swp}(v, V')$.

That is, it holds that $\mathbb{C} \cup \mathbb{B}(V') \vDash \mathit{All}^+(P^v)$ but $\mathbb{C} \cup \mathbb{B}(V') \nvDash \mathit{All}^+(P^v_*)$, the set V' is minimal with this property, and if V' is not simple below v, we remove vertices in a bottom-up fashion until we get such a set $V \subseteq V'$. The reader can verify that this definition matches the example in Figure 8.5.

Remark 8.14. Note that a black pebble on v is defined in terms of $\mathit{All}^+(P^v) = \bigvee_{u \in P^v} \bigvee_{i=1}^d u_i$ instead of just $\bigvee_{i=1}^d v_i$. Otherwise, we will not be able to prove the correspondence between L-pebblings and resolution derivation that we need. This means that if we let, say,

$$\mathbb{C}' = \left[\begin{array}{l|l} \overline{x}_i \lor \overline{v}_j \lor \bigvee_{n=1}^d z_n & \\ \overline{p}_i \lor \overline{q}_j \lor \bigvee_{n=1}^d r_n \lor \bigvee_{n=1}^d x_n & i,j \in [d] \\ \bigvee_{n=1}^d w_n \lor \bigvee_{n=1}^d z_n & \end{array} \right] \tag{8.1}$$

in Figure 8.5, then $\mathbb{C}'$ induces the same pebble subconfigurations as does $\mathbb{C}$, so $\mathbb{L}(\mathbb{C}') = \mathbb{L}(\mathbb{C}) = \{z\langle x, v\rangle, r\langle p, q\rangle, w\langle \emptyset \rangle\}$.

The reason we use $V = \mathit{swp}(v, V')$ instead of $V' \cap T^v_*$ (or even $V' \setminus P^v$) to define the white pebbles is that for technical purposes, we would like to have simple sets V below v in our induced subconfigurations $v\langle V\rangle$, but the minimal supporting sets V' do not necessarily have this property. For instance, in the clause configuration

$$\mathbb{C}'' = \left[\begin{array}{l|l} \overline{r}_i \lor \overline{x}_j \lor \overline{v}_l \lor \bigvee_{n=1}^d z_n & \\ \overline{p}_i \lor \overline{q}_j \lor \bigvee_{n=1}^d r_n \lor \bigvee_{n=1}^d x_n & i,j,l \in [d] \\ \overline{v}_l \lor \bigvee_{n=1}^d w_n \lor \bigvee_{n=1}^d z_n & \end{array} \right] \tag{8.2}$$

the vertices z and w have minimal supports $\{r, x, v\}$ and $\{v\}$, respectively, which are not simple sets below z and w, but since Definition 8.13 ignores all but the topmost vertices below the supported vertex, we get $\mathbb{L}(\mathbb{C}'') = \mathbb{L}(\mathbb{C}) = \{z\langle x, v\rangle, r\langle p, q\rangle, w\langle \emptyset \rangle\}$.

Thanks to this we get cleaner pebblings to work with (this will be used in Section 8.5), and it seems very plausible anyway that optimal resolution derivations should never result in clause configurations like $\mathbb{C}''$. Indeed, since the bound we will prove is asymptotically tight, we see that we do not really lose anything by restricting the white pebbles to $V = \mathit{swp}(v, V')$ instead of $V' \cap T^v_*$ or $V' \setminus P^v$.

Recall that the goal of this section is to demonstrate that resolution derivations induce L-pebblings. Suppose that $\pi = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ is a resolution derivation of $\bigvee_{i=1}^d z_i$ from $^*\mathit{Peb}^d_T$. For $\mathbb{C}_0 = \emptyset$ we have $\mathbb{L}(\mathbb{C}_0) = \emptyset$, and $\mathbb{C}_\tau = \{\bigvee_{i=1}^d z_i\}$ induces a single independent black pebble $\mathbb{L}(\mathbb{C}_\tau) = \{z\langle \emptyset \rangle\}$ on the root of T. Hence, we are done if we can show that $\{\mathbb{L}(\mathbb{C}_0), \ldots \mathbb{L}(\mathbb{C}_\tau)\}$ is a legal L-pebbling.

The rest of this section is devoted to proving that this is (almost) the case. We start by stating three technical lemmas. The first lemma relates subset containment of supporting sets and the order relation between corresponding subconfigurations.

Lemma 8.15. *For a vertex $v \in V(T)$, if $u \in P^v$ is a vertex and $U', V' \subseteq T \setminus P^v$ are vertex sets such that $U' \cap T^v_* \subseteq V' \cap T^v_*$, then $u\langle swp(u, U')\rangle \succeq v\langle swp(v, V')\rangle$.*

Proof. Let $U = swp(u, U')$ and $V = swp(v, V')$. According to Definition 8.4, we need to show that $cover(v\langle V\rangle) \subseteq cover(u\langle U\rangle)$.

Suppose $w \in cover(v\langle V\rangle)$. This means that there is a path $P_1 : w \rightsquigarrow v$ from w to v such that $P_1 \cap V = \emptyset$. Also, since $u \in P^v$ there is a path $P_2 : v \rightsquigarrow u$. Concatenating these paths, we get a path $P = P_1 \cup P_2$ from w to u. We claim that $P \cap U = \emptyset$. If this is true, we have $w \in cover(u\langle U\rangle)$ and thus $cover(v\langle V\rangle) \subseteq cover(u\langle U\rangle)$, and the lemma follows.

To prove the claim, note first that since $U \subseteq U' \subseteq T \setminus P^v$, it holds that $P_2 \cap U = \emptyset$. Suppose P_1 intersects U, and let $x \in P_1 \cap U$. By assumption, $x \notin V$ since $P_1 \cap V = \emptyset$. But $x \in U \subseteq U' \cap T^v_* \subseteq V' \cap T^v_*$, so Definition 8.12 tells us that the reason $x \notin V$ must be that $P^x_* \cap V' \cap T^v \neq \emptyset$. Let $y \in P^x_* \cap V' \cap T^v$ be the vertex closest to v. Looking at Definition 8.12 again, since $P^y_* \cap V' = \emptyset$ by construction, we have $y \in V$. But if so, $P_1 \cap V \neq \emptyset$. Contradiction. □

A second handy lemma is that if V' is not minimal or v is not maximal with respect to $\mathbb{C}$, this just means that $\mathbb{C}$ induces something stronger than the subconfiguration $v\langle swp(v, V')\rangle$.

Lemma 8.16. *If $\mathbb{C} \cup \mathbb{B}(V') \vDash All^+(P^v)$ for $V' \subseteq T \setminus P^v$, then there is an induced subconfiguration $u\langle U\rangle \in \mathbb{L}(\mathbb{C})$ such that $v\langle swp(v, V')\rangle \preceq u\langle U\rangle$.*

Proof. Minimize $U' \subseteq V'$ and then pick $u \in P^v$ as close to the root as possible so that $\mathbb{C} \cup \mathbb{B}(U') \vDash All^+(P^u)$. Set $U = swp(u, U')$ and use Lemma 8.15. □

The following easy lemma will be used repeatedly.

Lemma 8.17. *Suppose that C, D are clauses and $\mathbb{C}$ is a set of clauses. Then $\mathbb{C} \cup \{C\} \vDash D$ if and only if $\mathbb{C} \vDash \overline{a} \vee D$ for all $a \in Lit(C)$.*

Proof. Assume that $\mathbb{C} \cup \{C\} \vDash D$ and consider an assignment α such that $\alpha(\mathbb{C}) = 1$ and $\alpha(D) = 0$ (if there is no such α, then $\mathbb{C} \vDash D \subseteq \overline{a} \vee D$). Such an α must set all $\overline{a}$ to true. Conversely, if $\mathbb{C} \vDash \overline{a} \vee D$ for all $a \in Lit(C)$ and α is such that $\alpha(\mathbb{C}) = \alpha(C) = 1$, it must hold that $\alpha(D) = 1$, since otherwise $\alpha(\overline{a} \vee D) = 0$ for some literal $a \in Lit(C)$ satisfied by α. □

Using these lemmas, we can prove that resolution derivations induce L-pebblings. By the L-pebbling rules in Definition 8.5, any subconfiguration $v\langle V\rangle$ may be erased freely at any time. Consequently, we need not worry about subconfigurations disappearing during the transition from $\mathbb{C}_t$ to $\mathbb{C}_{t+1}$. What we do need to check, though, is that no $v\langle V\rangle$ appears inexplicably in $\mathbb{L}(\mathbb{C}_{t+1})$ as a result of a derivation step $\mathbb{C}_t \rightsquigarrow \mathbb{C}_{t+1}$, but that we can always derive any subconfiguration $v\langle V\rangle \in \mathbb{L}(\mathbb{C}_{t+1}) \setminus \mathbb{L}(\mathbb{C}_t)$ from $\mathbb{L}(\mathbb{C}_t)$ by the L-pebbling rules.

Let us consider the resolution derivation rules one by one.

Observation 8.18 (Inference). *If $\mathbb{C}_{t+1}$ is derived from $\mathbb{C}_t$ by inference, then $\mathbb{L}(\mathbb{C}_{t+1}) = \mathbb{L}(\mathbb{C}_t)$.*

Proof. This is immediate, since $\mathbb{C}_t$ and $\mathbb{C}_{t+1}$ imply exactly the same clauses. □

We remark that, as was stated in Section 3.1, this means that the exact definition of the resolution derivation rule is not important. The lower bound on space will hold for any sound derivation rule as long as the lines in the proof are disjunctive clauses.

Lemma 8.19 (Erasure). *Suppose that $\mathbb{C}_{t+1}$ is derived from $\mathbb{C}_t$ by erasure. Then for each $v\langle V\rangle \in \mathbb{L}(\mathbb{C}_{t+1})$ there is a $u\langle U\rangle \in \mathbb{L}(\mathbb{C}_t)$ such that $v\langle V\rangle \preceq u\langle U\rangle$.*

Proof. By assumption there is a $V' \subseteq T \setminus P^v$ such that $V = \mathit{swp}(v, V')$ and $\mathbb{C}_{t+1} \cup \mathbb{B}(V') \vDash \mathit{All}^+(P^v)$. Certainly, the same implication holds for $\mathbb{C}_t \supseteq \mathbb{C}_{t+1}$. The lemma follows from Lemma 8.16. □

In particular, all new subconfigurations resulting from an erasure $\mathbb{C}_t \rightsquigarrow \mathbb{C}_{t+1}$ can be obtained from $\mathbb{L}(\mathbb{C}_t)$ by reversal moves.

Lemma 8.20 (Axiom download). *If $\mathbb{C}_{t+1} = \mathbb{C}_t \cup \{C\}$ for an axiom clause $C \in \mathit{Ax}^d(r)$, then all subconfigurations $v\langle V\rangle \in \mathbb{L}(\mathbb{C}_{t+1}) \setminus \mathbb{L}(\mathbb{C}_t)$ can be obtained from $\mathbb{L}(\mathbb{C}_t) \cup r\langle \mathit{pred}(r)\rangle$ by reversals from subconfigurations in $\mathbb{L}(\mathbb{C}_t)$ followed by mergers on the vertices $\{r\} \cup \mathit{pred}(r)$.*

Proof. Let us fix a vertex $v \in V(T)$ and an axiom $C \in \mathit{Ax}^d(r)$. If $v\langle V\rangle$ is a pebble subconfiguration induced at time $t+1$, by assumption there is a minimal $V' \subseteq T \setminus P^v$ with $V = \mathit{swp}(v, V') \subseteq V'$ such that $\mathbb{C}_t \cup \{C\} \cup \mathbb{B}(V') \vDash \mathit{All}^+(P^v)$.

Our intuition is that downloading $C \in \mathit{Ax}^d(r)$ should not yield any interesting new subconfigurations $v\langle V\rangle$ if $r \in T \setminus T^v$, and for $r \in T^v$ we should be able to explain new subconfigurations with the help of an introduction of $r\langle \mathit{pred}(r)\rangle$ in our L-pebbling. We prove this by a case analysis over r.

$r \in T \setminus (T^v \cup P^v)$**:** Observing that $\mathbb{B}(r) \vDash C$ (this will be used repeatedly), we get that $\mathbb{C}_t \cup \mathbb{B}(V' \cup \{r\}) \vDash \mathit{All}^+(P^v)$ for $V' \cup \{r\} \subseteq T \setminus P^v$. Lemma 8.16 tells us that there is a $u\langle U\rangle \in \mathbb{L}(\mathbb{C}_t)$ such that $u\langle U\rangle \succeq v\langle \mathit{swp}(v, V' \cup \{r\})\rangle = v\langle \mathit{swp}(v, V')\rangle = v\langle V\rangle$, where the first equality follows since $r \notin T^v_*$. Hence, we can get $v\langle V\rangle$ from $\mathbb{L}(\mathbb{C}_t)$ by a reversal move.

$r \in P^v_*$**:** Write $C = \overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} r_l$ for $\{p, q\} = \mathit{pred}(r) \neq \emptyset$ and assume without loss of generality that p is the vertex in $P^v \cap \mathit{pred}(r)$. Using Lemma 8.17 to move p_i to the right of the implication sign yields $\mathbb{C}_t \cup \mathbb{B}(V') \vDash \mathit{All}^+(P^v) \vee p_i = \mathit{All}^+(P^v)$, and since V' is minimal it follows that $v\langle V\rangle \in \mathbb{L}(\mathbb{C}_t)$.

$r = v$: Note first that the introduction of $r\langle \mathit{pred}(r)\rangle$ is a legal pebbling move, so if $\mathbb{C}_t \cup \{C\} \cup \mathbb{B}(V') \vDash \mathit{All}^+(P^r)$ for $\mathit{pred}(r) \subseteq V'$, no further analysis is needed for $r\langle \mathit{swp}(r, V')\rangle = r\langle \mathit{pred}(r)\rangle$. In particular, this is always the case if $\mathit{pred}(r) = \emptyset$, i.e., if r is a source.

Suppose that $v\langle V\rangle = r\langle \mathit{swp}(r, V')\rangle \in \mathbb{L}(\mathbb{C}_{t+1})$ for $V \neq \mathit{pred}(r) = \{p, q\}$, and write $C = \overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} r_l$. We want to derive $r\langle V\rangle$ by the pebbling rules from $\mathbb{L}(\mathbb{C}_t) \cup \bigl\{r\langle \mathit{pred}(r)\rangle\bigr\}$. By symmetry, we get two subcases.

1. $p \in V, q \notin V$: By Definition 8.12, we have $p \in V' \supseteq V$. Also, it must hold that $q \notin V'$, since otherwise $P^q_* \cap V' \subseteq P^q_* \cap (T \setminus P^r) = P^q_* \cap (T \setminus P^q_*) = \emptyset$ would imply that $q \in V = \mathit{swp}(v, V')$, contrary to assumption. It follows that $V' \subseteq T \setminus P^q$. Also, we can use Lemma 8.17 to move q_j to the right-hand side of the implication sign and get $\mathbb{C}_t \cup \mathbb{B}(V') \vDash \mathit{All}^+(P^r) \vee q_j \subseteq \mathit{All}^+(P^r) \vee \bigvee_{l=1}^{d} q_l = \mathit{All}^+(P^q)$. Plugging this into Lemma 8.16 shows that there is a $w\langle W\rangle \in \mathbb{L}(\mathbb{C}_t)$ such that $q\bigl\langle V \setminus \{p\}\bigr\rangle = q\bigl\langle \mathit{swp}(q, V')\bigr\rangle \preceq w\langle W\rangle$. Thus we can derive $q\bigl\langle V \setminus \{p\}\bigr\rangle$ from $\mathbb{L}(\mathbb{C}_t)$ by reversal and then merge $r\langle \mathit{pred}(r)\rangle = r\langle p, q\rangle$ with $q\bigl\langle V \setminus \{p\}\bigr\rangle$ to obtain $r\bigl\langle \bigl(\{p, q\} \cup (V \setminus \{p\})\bigr) \setminus \{q\}\bigr\rangle = r\langle V\rangle$.
2. $p, q \notin V$: Again, by Definition 8.12 we have $p, q \notin V'$. If we use Lemma 8.17 twice, we get $\mathbb{C}_t \cup \mathbb{B}(V') \vDash \mathit{All}^+(P^p) \wedge \mathit{All}^+(P^q)$, and noting that $V' \subseteq T \setminus \bigl(P^p \cup P^q\bigr)$ we can apply Lemma 8.16 to derive $p\bigl\langle V \cap T^p_*\bigr\rangle$ and $q\bigl\langle V \cap T^q_*\bigr\rangle$ from $\mathbb{L}(\mathbb{C}_t)$ by reversal moves. Merging these pebble subconfigurations with $r\langle p, q\rangle$, we get the desired pebble subconfiguration $r\bigl\langle \bigl(V \cap T^p_*\bigr) \cup \bigl(V \cap T^q_*\bigr)\bigr\rangle = r\langle V\rangle$.

We note in passing that this is the place in the proof where we critically need black pebbles to be defined in terms of $\mathit{All}^+(P^v) = \bigvee_{u \in P^v} \bigvee_{i=1}^{d} u_i$ instead of just $\bigvee_{i=1}^{d} v_i$. (Although it also simplifies the proof of the case $r \in P^v_*$, there it is not strictly necessary.)

$r \in T^v_*$: By assumption, $\mathbb{C}_t \cup \{C\} \cup \mathbb{B}(V') \vDash \mathit{All}^+(P^v)$, and since $r \in T^v_*$ and $\mathbb{B}(r) \vDash C$ we have $\mathbb{C}_t \cup \mathbb{B}(V' \cup \{r\}) \vDash \mathit{All}^+(P^v)$ for $V' \cup \{r\} \subseteq T \setminus P^v$. If $P^r \cap V' \neq \emptyset$, it holds that $\mathit{swp}(v, V' \cup \{r\}) = \mathit{swp}(v, V')$, and we can obtain $v\langle V\rangle$ from $\mathbb{L}(\mathbb{C}_t)$ by reversal according to Lemma 8.16. Suppose therefore that $P^r \cap V' = \emptyset$. Also, we assume that $\mathbb{C}_t \cup \mathbb{B}(V') \nvDash \mathit{All}^+(P^v)$ since otherwise $v\langle V\rangle \in \mathbb{L}(\mathbb{C}_t)$ and there is nothing to prove.

Pick $U' \subseteq V' \cup \{r\}$ minimal and then $u \in P^v$ maximal with respect to U' such that $\mathbb{C}_t \cup \mathbb{B}(U') \vDash \mathit{All}^+(P^u)$. Since $\mathbb{C}_t \cup \mathbb{B}(V') \nvDash \mathit{All}^+(P^v)$ we must have $r \in U'$. Set $U = \mathit{swp}(u, U')$. Using that $P^r_* \cap U' \subseteq P^r_* \cap V' = \emptyset$, we see that $r \in U$. Consequently, we cannot use $u\langle U\rangle \in \mathbb{L}(\mathbb{C}_t)$ to derive $v\langle V\rangle \npreceq u\langle U\rangle$ by reversal. However, since $U' \subseteq V' \cup \{r\}$, Lemma 8.15 says that $v\bigl\langle (V \cup \{r\}) \setminus T^r_*\bigr\rangle = v\bigl\langle \mathit{swp}(v, V' \cup \{r\})\bigr\rangle \preceq u\langle U\rangle$ can be derived by reversal from $\mathbb{L}(\mathbb{C}_t)$. If we could also derive $r\big\langle V \cap T^r_*\big\rangle$ from $\mathbb{L}(\mathbb{C}_t) \cup \big\{r\langle pred(r)\rangle\big\}$, we could do a merger to get $v\big\langle\big(((V \cup \{r\}) \setminus T^r_*) \cup (V \cap T^r_*)\big) \setminus \{r\}\big\rangle = v\langle V\rangle$. Hence, we are done if we can derive the pebble subconfiguration $r\big\langle V \cap T^r_*\big\rangle = r\big\langle swp(v, V') \cap T^r_*\big\rangle = r\big\langle swp(r, V')\big\rangle$ from $\mathbb{L}(\mathbb{C}_t) \cup \big\{r\langle pred(r)\rangle\big\}$. But $All^+(P^r) \supseteq All^+(P^v)$, so by assumption we have $\mathbb{C}_t \cup \{C\} \cup \mathbb{B}(V') \vDash All^+(P^r)$ for $V' \subseteq T \setminus P^r$. This is almost exactly the case $r = v$ above, where we proved that $r\big\langle swp(r, V')\big\rangle$ is derivable from $\mathbb{L}(\mathbb{C}_t) \cup \big\{r\langle pred(r)\rangle\big\}$. The only difference is that now it is not necessarily true that V' is a minimal support and that r is maximal with respect to V'. But these assumptions were not used in the derivation of $r\big\langle swp(r, V')\big\rangle$ from $\mathbb{L}(\mathbb{C}_t) \cup \big\{r\langle pred(r)\rangle\big\}$ anyway, so we can reuse exactly the same proof to get $r\big\langle swp(r, V')\big\rangle$. This concludes the analysis for $r \in T^v_*$.

Studying the pebbling moves performed in the case analysis above, we see that all subconfigurations $v\langle V\rangle \in \mathbb{L}(\mathbb{C}_{t+1}) \setminus \mathbb{L}(\mathbb{C}_t)$ resulting from an axiom download can be obtained from $\mathbb{L}(\mathbb{C}_t) \cup r\langle pred(r)\rangle$ by a (possibly empty) sequence of reversals from $\mathbb{L}(\mathbb{C}_t)$, followed by a (possibly empty) sequence of mergers on $\{r\} \cup pred(r)$. □

Combining the results proven for axiom download, inference, and erasure, we can show that a resolution derivation induces a legal L-pebbling. We need a pair of easy technical observations about L-pebbling cost, which we state as a separate proposition for clarity.

Proposition 8.21. *For $\mathbb{L}_1$ and $\mathbb{L}_2$ arbitrary L-configurations, it holds that*

1. *if $\mathbb{L}_1 \subseteq \mathbb{L}_2$ then $\mathit{cost}(\mathbb{L}_1) \leq \mathit{cost}(\mathbb{L}_2)$, and*
2. *$\mathit{cost}(\mathbb{L}_1 \cup \mathbb{L}_2) \leq \mathit{cost}(\mathbb{L}_1) + \mathit{cost}(\mathbb{L}_2)$.*

Proof. This is fairly obvious, but we give a short formal proof for completeness. According to Definition 8.5, if $Bl(\mathbb{L}_1) \cup Wh(\mathbb{L}_1) \subseteq Bl(\mathbb{L}_2) \cup Wh(\mathbb{L}_2)$, then $\mathit{cost}(\mathbb{L}_1) = \big|Bl(\mathbb{L}_1) \cup Wh(\mathbb{L}_1)\big| \leq \big|Bl(\mathbb{L}_2) \cup Wh(\mathbb{L}_2)\big| = \mathit{cost}(\mathbb{L}_2)$. Part 1 follows immediately from this observation. Part 2 also follows easily, since each pebbled vertex on the left-hand side is counted at least once on the right-hand side. □

Theorem 8.22. *Let $\pi = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ be a resolution derivation of $\bigvee_{l=1}^d z_l$ from *Peb^d_T. Then the L-configurations $\mathbb{L}(\mathbb{C}_0), \ldots, \mathbb{L}(\mathbb{C}_\tau)$ are contained in a legal, complete L-pebbling $\mathcal{L}$ of T such that $\max_{t\in[\tau]}\big\{\mathit{cost}(\mathbb{L}(\mathbb{C}_t))\big\} = \Omega\big(\mathit{cost}(\mathcal{L})\big)$.*

Proof. The fact that $\big\{\mathbb{L}(\mathbb{C}_0), \ldots, \mathbb{L}(\mathbb{C}_\tau)\big\}$ is the "backbone" of a legal L-pebbling was proven in Observation 8.18, Lemma 8.19, and Lemma 8.20, where it was explicitly indicated how the "holes" in $\mathbb{L}(\mathbb{C}_t) \rightsquigarrow \mathbb{L}(\mathbb{C}_{t+1})$ could be filled in by L-pebbling moves to get a legal pebbling $\mathcal{L}$. It was also noted above that $\mathbb{L}(\mathbb{C}_0) = \emptyset$ and $\mathbb{L}(\mathbb{C}_\tau) = \big\{z\langle\emptyset\rangle\big\}$, so filling in the holes results in a complete pebbling of T.

The bound $\max_{t\in[\tau]}\big\{\mathit{cost}(\mathbb{L}(\mathbb{C}_t))\big\} = \Omega\big(\mathit{cost}(\mathcal{L})\big)$ does not follow immediately from this, however. The problem is that a single derivation step $\mathbb{C}_t \rightsquigarrow \mathbb{C}_{t+1}$ may induce several L-pebbling moves to get from $\mathbb{L}(\mathbb{C}_t)$ to $\mathbb{L}(\mathbb{C}_{t+1})$ in $\mathcal{L}$. Therefore, we have to consider the possibility[1] that the maximal pebbling cost in $\mathcal{L}$ is reached in some intermediate L-configuration $\mathbb{L}'$ between $\mathbb{L}(\mathbb{C}_t)$ and $\mathbb{L}(\mathbb{C}_{t+1})$.

Since inference steps in π do not change the set of induced L-configurations, we get two cases.

1. $\mathbb{C}_t \rightsquigarrow \mathbb{C}_{t+1}$ is an erasure. The moves to get from $\mathbb{L}(\mathbb{C}_t)$ to $\mathbb{L}(\mathbb{C}_{t+1})$ are a series of reversals from $\mathbb{L}(\mathbb{C}_t)$ followed by a series of erasures from $\mathbb{L}(\mathbb{C}_t)$. In view of part 1 of Proposition 8.21, the maximal cost is incurred in the intermediate L-configuration $\mathbb{L}'$ after all reversals but before all erasures. We have $\mathbb{L}' = \mathbb{L}(\mathbb{C}_t) \cup \mathbb{L}(\mathbb{C}_{t+1})$, and by part 2 of Proposition 8.21 it follows that $\mathit{cost}(\mathbb{L}') \leq \mathit{cost}(\mathbb{L}(\mathbb{C}_t)) + \mathit{cost}(\mathbb{L}(\mathbb{C}_{t+1})) \leq 2 \cdot \max_{i\in[t,t+1]}\bigl\{\mathit{cost}(\mathbb{L}(\mathbb{C}_i))\bigr\}$.

2. $\mathbb{C}_t \rightsquigarrow \mathbb{C}_{t+1}$ is a download of $C \in \mathit{Ax}^d(v)$. In this case the sequence of moves to get from $\mathbb{L}(\mathbb{C}_t)$ to $\mathbb{L}(\mathbb{C}_{t+1})$ is a possible introduction of $v\langle \mathit{pred}(v)\rangle$ followed by a series of reversals from $\mathbb{L}(\mathbb{C}_t)$, then a series of mergers on $\{v\} \cup \mathit{pred}(v)$, and finally a series of erasures of subconfigurations not derived in the merger moves. Again by part 1 of Proposition 8.21, we may concentrate on the L-configuration $\mathbb{L}'$ after all reversals and mergers but before the erasures.

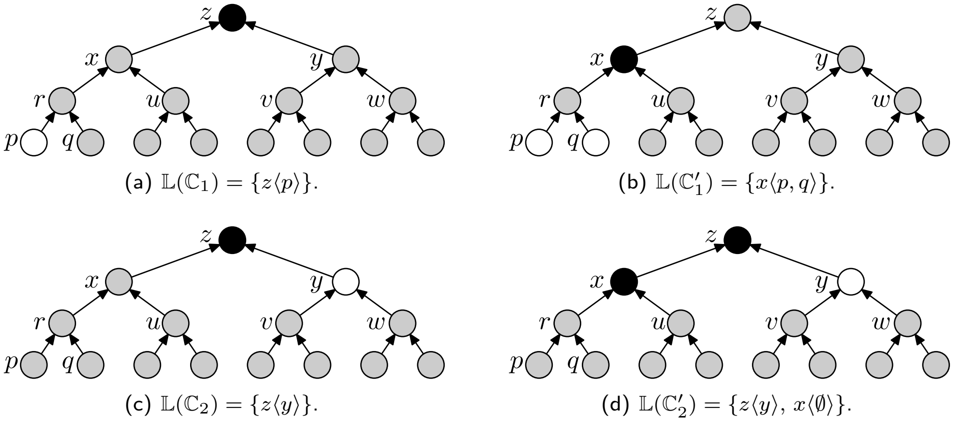
   All pebbles in $\mathit{Bl}(\mathbb{L}') \cup \mathit{Wh}(\mathbb{L}')$ are present in either $\mathbb{L}(\mathbb{C}_t)$ or $\mathbb{L}(\mathbb{C}_{t+1})$, except possibly for the pebbles on $\{v\} \cup \mathit{pred}(v)$ which may have been introduced and then merged away. Since by construction all subconfigurations resulting from these mergers must be contained in $\mathbb{L}(\mathbb{C}_{t+1})$, the pebbles on $\{v\} \cup \mathit{pred}(v)$ are the only ones that can appear and then disappear during the intermediate pebbling steps. If we remove $\{v\} \cup \mathit{pred}(v)$ from $\mathit{Bl}(\mathbb{L}') \cup \mathit{Wh}(\mathbb{L}')$, the pebbling cost cannot decrease by more than 3.

   Since all pebbles $\mathit{Bl}(\mathbb{L}') \setminus \bigl(\{v\} \cup \mathit{pred}(v)\bigr)$ and $\mathit{Wh}(\mathbb{L}') \setminus \bigl(\{v\} \cup \mathit{pred}(v)\bigr)$ are contained in $\mathit{Bl}(\mathbb{L}(\mathbb{C}_t)) \cup \mathit{Bl}(\mathbb{L}(\mathbb{C}_{t+1}))$ and $\mathit{Wh}(\mathbb{L}(\mathbb{C}_t)) \cup \mathit{Wh}(\mathbb{L}(\mathbb{C}_{t+1}))$, respectively, appealing to part 2 of Proposition 8.21 again we get the inequality $\max_{i\in[t,t+1]}\bigl\{\mathit{cost}(\mathbb{L}(\mathbb{C}_i))\bigr\} \geq \frac{1}{2}\bigl(\mathit{cost}(\mathbb{L}') - 3\bigr)$.

This establishes that even if the maximal cost in the L-pebbling $\mathcal{L}$ induced by derivation $\pi = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ is attained in some intermediate L-configuration $\mathbb{L}' \notin \bigl\{\mathbb{L}(\mathbb{C}_t) \mid t \in [\tau]\bigr\}$, it still holds that $\max_{t\in[\tau]}\bigl\{\mathit{cost}(\mathbb{L}(\mathbb{C}_t))\bigr\} \geq \frac{1}{2}\mathit{cost}(\mathcal{L}) + \mathrm{O}(1)$. The theorem follows. □

Remark 8.23. At this point, the reader might ask whether we really need the reversal rule in the L-pebble game in order to get Theorem 8.22 or whether it is just a convenience to simplify the proofs. The answer is that unfortunately, the reversal rule is really needed. We provide two examples of this below, using the binary tree of height 3 with vertex labels as in Figure 8.6.

[1] In fact, this does not happen, but instead of proving this we happily sacrifice a constant 2 here in order to get a simpler (or at least slightly less involved) proof.



(a) $\mathbb{L}(\mathbb{C}_1) = \{z\langle p\rangle\}$.

(b) $\mathbb{L}(\mathbb{C}'_1) = \{x\langle p, q\rangle\}$.

(c) $\mathbb{L}(\mathbb{C}_2) = \{z\langle y\rangle\}$.

(d) $\mathbb{L}(\mathbb{C}'_2) = \{z\langle y\rangle,\, x\langle\emptyset\rangle\}$.

Figure 8.6: Illustration of reversal moves in Remark 8.23.

Suppose that we have

$$\mathbb{C}_1 = \left[\begin{array}{l|l} \overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} x_l & \multirow{2}{*}{$i, j \in [d]$} \\ \overline{p}_i \vee \bigvee_{l=1}^{d} z_l & \end{array}\right] \quad \text{with} \quad \mathbb{L}(\mathbb{C}_1) = \bigl\{z\langle p\rangle\bigr\} \tag{8.3}$$

(see Figure 8.6(a)). Note that only the subset of clauses on the second line in $\mathbb{C}_1$ contributes to $\mathbb{L}(\mathbb{C}_1)$. It is true that, because of the clauses on the first line, we have

$$\mathbb{C}_1 \cup \mathbb{B}(p, q) \vDash All^{+}(P^{x}) = \bigvee_{l=1}^{d} x_l \vee \bigvee_{l=1}^{d} z_l \ , \tag{8.4}$$

but the support $\{p, q\}$ is not minimal and x is not maximal with respect to $\mathbb{C}_1$ and $\{p, q\}$ (Definition 8.12) since it also holds that

$$\mathbb{C}_1 \cup \mathbb{B}(p) \vDash All^{+}(P^{x}_{*}) = \bigvee_{l=1}^{d} z_l \ . \tag{8.5}$$

However, if we erase the second line of clauses from (8.3), the implication in (8.4) comes into play, and we get

$$\mathbb{C}'_1 = [\overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} x_l \,|\, i \in [d]] \quad \text{with} \quad \mathbb{L}(\mathbb{C}'_1) = \bigl\{x\langle p, q\rangle\bigr\} \tag{8.6}$$

as in Figure 8.6(b). It is necessary to have the reversal rule to go from Figure 8.6(a) to Figure 8.6(b), which shows why reversals are needed in Lemma 8.19.

This might perhaps look like a somewhat silly example, but it nevertheless pinpoints the problem: although the erasures going from $\mathbb{C}_1$ in (8.3) to $\mathbb{C}'_1$ in (8.6) might seem clearly non-optimal, we cannot exclude the possibility that such derivation steps are made, and so we have to be able to match such steps by pebbling moves.

As a second example, consider

$$\mathbb{C}_2 = \left[\begin{array}{l|l} x_1 \lor v_1 \lor \bigvee_{j=1}^{d} z_j & \\ x_1 \lor w_1 \lor \bigvee_{j=1}^{d} z_j & i \in [d] \\ \overline{y}_i \lor \bigvee_{j=1}^{d} z_j & \end{array} \right] \quad \text{with} \quad \mathbb{L}(\mathbb{C}_2) = \bigl\{ z\langle y \rangle \bigr\} \tag{8.7}$$

(see Figure 8.6(c)). Here the first two clauses do not contribute to $\mathbb{L}(\mathbb{C}_2)$, but if we download the axiom $\overline{v}_1 \lor \overline{w}_1 \lor \bigvee_{j=1}^{d} y_j$, we get

$$\mathbb{C}_2' = \left[\begin{array}{l|l} x_1 \lor v_1 \lor \bigvee_{j=1}^{d} z_j & \\ x_1 \lor w_1 \lor \bigvee_{j=1}^{d} z_j & \\ \overline{y}_i \lor \bigvee_{j=1}^{d} z_j & i \in [d] \\ \overline{v}_1 \lor \overline{w}_1 \lor \bigvee_{j=1}^{d} y_j & \end{array} \right] \quad \text{with} \quad \mathbb{L}(\mathbb{C}_2') = \bigl\{ z\langle y \rangle, x\langle \emptyset \rangle \bigr\} \tag{8.8}$$

as in Figure 8.6(d), since it is easy to check that $\mathbb{C}_2' \vDash \mathit{All}^{+}(P^{x}) = \bigvee_{j=1}^{d} x_j \lor \bigvee_{j=1}^{d} z_j$ but $\mathbb{C}_2' \nvDash \mathit{All}^{+}(P^{x}_{*}) = \bigvee_{j=1}^{d} z_j$. We cannot get $x\langle \emptyset \rangle$ from $\mathbb{L}(\mathbb{C}_2)$ unless we have reversal moves, so the reversal rule is needed also in Lemma 8.20.

We leave it to the reader to verify that $\mathbb{C}_1$ and $\mathbb{C}_2$ can indeed be derived from $^{*}\mathit{Peb}^{d}_{T_3}$. We note, though, that it appears that in order to derive $\mathbb{C}_2$ one needs to pass stronger clause configurations along the way, and it seems very unclear why anyone would like to go from these clause configurations to the weaker configuration $\mathbb{C}_2$.

We conclude this section by proving Theorem 8.1 on page 98. Since we wanted to avoid unnecessary technicalities in Section 8.1, Theorem 8.1 talks about refutations $\pi : \mathit{Peb}^{d}_{T_h} \vdash 0$ rather than derivations $\pi^{*} : {}^{*}\mathit{Peb}^{d}_{T_h} \vdash \bigvee_{i=1}^{d} z_i$, but this is easily taken care of.

Theorem 8.1 (restated). *There is a translation function from clause configurations derived from $\mathit{Peb}^{d}_{T_h}$ into L-configurations in T_h such that any resolution refutation π of $\mathit{Peb}^{d}_{T_h}$ corresponds to a complete labelled pebbling $\mathcal{L}_{\pi}$ of T_h under this translation.*

Proof. Given a resolution refutation $\pi : \mathit{Peb}^{d}_{T_h} \vdash 0$, use (the proof of) Lemma 8.9 to transform the refutation π clause configuration by clause configuration into a derivation $\pi^{*} : {}^{*}\mathit{Peb}^{d}_{T_h} \vdash \bigvee_{i=1}^{d} z_i$ in the same space. Then use Definition 8.13 as the translation function, and let $\mathcal{L}_{\pi}$ be the labelled pebbling constructed from π^{*} in Theorem 8.22. □

We comment that as another attempt to simplify the exposition in Section 8.1, Theorem 8.1 leaves out the crucial information in Theorem 8.22 that the cost of $\mathcal{L}_{\pi}$ is upper-bounded by the maximal cost of the induced L-configurations $\mathbb{L}(\mathbb{C}_t)$. We will return to Theorem 8.22 and use this information in the proof of Theorem 8.2 at the end of the next section.

8.4 Induced L-pebbles Measure Clause Set Size

In the last section, we proved that $Sp(Peb^d_{T_h} \vdash 0) = Sp({}^*Peb^d_{T_h} \vdash \bigvee_{i=1}^d z_i)$ and that each resolution derivation $\pi : {}^*Peb^d_{T_h} \vdash \bigvee_{i=1}^d z_i$ induces a complete L-pebbling $\mathcal{L}$ of T_h such that $\max_{\mathbb{C} \in \pi}\{\mathit{cost}(\mathbb{L}(\mathbb{C}))\} = \Omega(\mathit{cost}(\mathcal{L}))$. In Section 8.2 we stated (promising a proof in Section 8.5) that $\mathit{cost}(\mathcal{L}) = \Omega(\mathit{BW\text{-}Peb}(T))$. The final component needed to piece together the proof of our lower bound on the refutation space of pebbling contradictions is that the number of pebbles in an induced L-configuration $\mathbb{L}(\mathbb{C})$ and the number of clauses in $\mathbb{C}$ are somehow connected.

Note that we cannot expect a proof of this fact to work regardless of the pebbling degree d. The induced L-pebbling in Section 8.3 makes no assumptions about d, but we know that $Sp({}^*Peb^1_G \vdash z_1) = Sp(Peb^1_G \vdash 0) = \mathrm{O}(1)$. If we look at the resolution refutation π of Peb^1_G in constant space sketched in Section 5.2, we see that the induced L-pebbling starts by placing white pebbles on $\mathit{pred}(z)$ and a black pebble on z, i.e., introducing $z\langle \mathit{pred}(z)\rangle$, and then pushes the white pebbles downwards by introducing $v\langle \mathit{pred}(v)\rangle$ for all v in reverse topological order and merging until it reaches $z\langle S\rangle$ for S the source vertices of G. Finally, the white pebbles $s \in S$ are eliminated one by one by introducing $s\langle \emptyset\rangle$ and merging. The reason that Peb^1_G can be refuted in constant space is that one single clause $z_1 \vee \bigvee_{v \in V} \overline{v}_1$ can induce an arbitrary number $|V|$ of white pebbles, or, phrasing it differently, that white pebbles are free for $d = 1$.

In Theorem 8.29 below we show that provided $d > 1$ one has to pay at least $|\mathbb{C}| \geq N$ clauses to get N induced pebbles. This completes the proof of our main theorem which was outlined in Section 8.1. We first show some technical results about CNF formulas that will be needed in the proof.

Lemma 8.24. *Suppose that it holds for a set of clauses $\mathbb{C}$ and clauses D_1 and D_2 with $\mathit{Vars}(D_1) \cap \mathit{Vars}(D_2) = \emptyset$ that $\mathbb{C} \vDash D_1 \vee D_2$ but $\mathbb{C} \nvDash D_2$. Then there is a literal $a \in \mathit{Lit}(\mathbb{C}) \cap \mathit{Lit}(D_1)$.*

Proof. Pick a truth value assignment α such that $\alpha(\mathbb{C}) = 1$ but $\alpha(D_2) = 0$. Since $\mathbb{C} \vDash D$, we must have $\alpha(D_1) = 1$. Let α' be the same assignment except that all satisfied literals in D_1 are flipped to false (which is possible since they are all strictly distinct by assumption). Then $\alpha'(D_1 \vee D_2) = 0$ forces $\alpha'(\mathbb{C}) = 0$, so the flip must have falsified some previously satisfied clause in $\mathbb{C}$. □

Definition 8.25. A set of clauses $\mathbb{C}$ implies a clause D *minimally* if $\mathbb{C} \vDash D$ but for all $\mathbb{C}' \subsetneqq \mathbb{C}$ it holds that $\mathbb{C}' \nvDash D$. If $\mathbb{C} \vDash 0$ minimally, $\mathbb{C}$ is said to be *minimally unsatisfiable.*

Lemma 8.26. *Let $\mathbb{C}$ be a set of clauses and D a clause such that $\mathbb{C} \vDash D$ minimally and $a \in \mathit{Lit}(\mathbb{C})$ but $\overline{a} \notin \mathit{Lit}(\mathbb{C})$. Then $a \in \mathit{Lit}(D)$.*

Proof. Suppose not. Let $\mathbb{C}_1 = \{C \in \mathbb{C} \mid a \in \mathit{Lit}(C)\}$ and $\mathbb{C}_2 = \mathbb{C} \setminus \mathbb{C}_1$. Since $\mathbb{C}_2 \nvDash D$ there is a truth value assignment α such that $\alpha(\mathbb{C}_2) = 1$ and $\alpha(D) = 0$.

Note that $\alpha(a) = 0$, since otherwise $\alpha(\mathbb{C}_1) = 1$ which would contradict $\mathbb{C}_1 \cup \mathbb{C}_2 = \mathbb{C} \vDash D$. It follows that $\overline{a} \notin \mathit{Lit}(D)$. Flip a to true and denote the resulting truth value assignment by $\alpha^{a=1}$. By construction $\alpha^{a=1}(\mathbb{C}_1) = 1$ and $\mathbb{C}_2$ and D are not affected since $\{a, \overline{a}\} \cap \big(\mathit{Lit}(\mathbb{C}_2) \cup \mathit{Lit}(D)\big) = \emptyset$, so $\alpha^{a=1}(\mathbb{C}) = 1$ and $\alpha^{a=1}(D) = 0$. Contradiction. □

The fact that a minimally unsatisfiable CNF formula must have more clauses than variables seems to have been proven independently a number of times (see, for instance, [2, 11, 29, 53]). We will need the following formulation of this result, relating subsets of variables in a minimally implicating CNF formula and the clauses containing variables from these subsets.

Theorem 8.27. *Suppose that F is a CNF formula that implies a clause D minimally. For any subset of variables V, let $F_V = \{C \in F \mid \mathit{Vars}(C) \cap V \neq \emptyset\}$. Then if $V \subseteq \mathit{Vars}(F) \setminus \mathit{Vars}(D)$, it holds that $|F_V| > |V|$. In particular, if F is minimally unsatisfiable, we have $|F_V| > |V|$ for all $V \subseteq \mathit{Vars}(F)$.*

Proof. The proof is by induction over $V \subseteq \mathit{Vars}(F) \setminus \mathit{Vars}(D)$.

If $|V| = 1$, then $|F_V| \geq 2$, since any $x \in V$ must occur both positively and negatively in F by Lemma 8.26.

The inductive step just generalizes the proof of Lemma 8.26. Suppose that $|F_{V'}| > |V'|$ for all strict subsets $V' \subsetneqq V \subseteq \mathit{Vars}(F) \setminus \mathit{Vars}(D)$ and consider V. Since $F_{V'} \subseteq F_V$ if $V' \subseteq V$, choosing any V' of size $|V| - 1$ we see that $|F_V| \geq |F_{V'}| \geq |V'| + 1 = |V|$.

If $|F_V| > |V|$ there is nothing to prove, so assume that $|F_V| = |V|$. Consider the bipartite graph with the variables V and the clauses in F_V as vertices, and edges between variables and clauses for all variable occurrences. Since for all $V' \subseteq V$ the set of neighbours $N(V') = F_{V'} \subseteq F_V$ satisfies $|N(V')| \geq |V'|$, by Hall's marriage theorem there is a perfect matching between V and F_V. Use this matching to satisfy F_V assigning values to variables in V only.

The clauses in $F' = F \setminus F_V$ are not affected by this partial truth value assignment, since they do not contain any occurrences of variables in V. Furthermore, by the minimality of F it must hold that F' can be satisfied and D falsified simultaneously by assigning values to variables in $\mathit{Vars}(F') \setminus V$.

The two partial truth value assignments above can be combined to an assignment that satisfies all of F but falsifies D, which is a contradiction. Thus $|F_V| > |V|$. The theorem follows by induction. □

We need one final definition relating vertices of T and literal occurrences in clauses for the variables associated with these vertices.

Definition 8.28. We say that a vertex v is *represented positively* in a clause C if $\{v_1, \ldots, v_d\} \cap \mathit{Lit}(C) \neq \emptyset$ and *negatively* if $\{\overline{v}_1, \ldots, \overline{v}_d\} \cap \mathit{Lit}(C) \neq \emptyset$, and that C *mentions v positively* or *negatively*, respectively. This definition is extended to sets of vertices and clauses by taking unions.

For a set of vertices U, we let $\mathit{Vars}^d(U) = \{u_1, \ldots, u_d \mid u \in U\}$ denote the set of all variables representing vertices in U. For a set of clauses $\mathbb{C}$, we use $V(\mathbb{C}) = \{u \in U \mid \mathit{Vars}^d(u) \cap \mathit{Vars}(\mathbb{C}) \neq \emptyset\}$ to denote all vertices represented (positively or negatively) in $\mathbb{C}$, and we write $\mathbb{C}[\![U]\!] = \{C \in \mathbb{C} \mid V(C) \cap U \neq \emptyset\}$ to denote the subset of all clauses in $\mathbb{C}$ mentioning vertices in U.

We now prove by induction over the (sub)sets of induced pebbles that a clause configuration is at least as large as the number of pebbles it induces.

Theorem 8.29. *Suppose that $\mathbb{C}$ is a set of clauses derived from $^*\mathit{Peb}^d_T$ for $d \geq 2$ that induces the labelled pebble configuration $\mathbb{L}(\mathbb{C})$. Then $\mathsf{cost}(\mathbb{L}(\mathbb{C})) \leq |\mathbb{C}|$.*

Proof. Suppose that $\mathbb{C}$ induces a subconfiguration $v\langle W\rangle$. By Definition 8.13, there is a minimal support $V_v \subseteq T \setminus P^v$ with $W = \mathit{swp}(v, V_v) \subseteq V_v$ such that $\mathbb{C} \cup \mathbb{B}(V_v) \vDash \mathit{All}^+(P^v)$ but $\mathbb{C} \cup \mathbb{B}(V_v) \nvDash \mathit{All}^+(P^v_*)$ and $\mathbb{C} \cup \mathbb{B}(V'_v) \nvDash \mathit{All}^+(P^v)$ for all $V'_v \subsetneqq V_v$.

Fix for each induced subconfiguration $v\langle W\rangle$ with $W = \mathit{swp}(v, V_v)$ a subset $\mathbb{C}_v \subseteq \mathbb{C}$ such that the implication $\mathbb{C}_v \cup \mathbb{B}(V_v) \vDash \mathit{All}^+(P^v)$ holds minimally. Since $\mathit{Vars}(\mathbb{B}(V_v)) \cap \mathit{Vars}(\mathit{All}^+(P^v)) = \emptyset$ by definition, using Lemma 8.24 with $D_1 = \bigvee_{i=1}^d v_i$ and $D_2 = \mathit{All}^+(P^v_*)$, we see that the vertex v must be represented in $\mathbb{C}_v$ by some positive literal v_i. For the white pebbles in $W \subseteq V_v$, it follows for the same reason from Lemma 8.26 that all literals $\overline{w}_j$, $j \in [d]$, must be present in $\mathbb{C}_v$.

We prove by induction over $U \subseteq \mathit{Bl}(\mathbb{L}(\mathbb{C})) \cup \mathit{Wh}(\mathbb{L}(\mathbb{C}))$ that $|\mathbb{C}[\![U]\!]| \geq |U|$, from which the theorem clearly follows. The base case $|U| = 1$ is immediate, since we just observed that all pebbled vertices $v \in V$ are represented in $\mathbb{C}$.

For the induction step, suppose that $\left|\mathbb{C}[\![U']\!]\right| \geq \left|U'\right|$ for all $U' \subsetneqq U$. Pick a "topmost" vertex $u \in U$, i.e., such that $P^u_* \cap U = \emptyset$, and look at the subconfiguration $v\langle W\rangle$ containing u (with $u = v$ if u is black and u strictly below v otherwise) and the associated subset $\mathbb{C}_v \subseteq \mathbb{C}$ fixed above. Note that $\mathit{Vars}^d(U) \cap \mathit{Vars}(\mathit{All}^+(P^v)) \subseteq \{u_1, \ldots, u_d\}$. Let $S = U \cap V(\mathbb{C}_v)$ be the set of all vertices in U mentioned by $\mathbb{C}_v$. We claim that $|\mathbb{C}_v[\![S]\!]| \geq |S|$.

To show this, note first that $u \in S$ as was argued above, and if $S = \{u\}$ we trivially have $|\mathbb{C}_v[\![S]\!]| \geq 1 = |S|$. Suppose therefore that $S \supsetneqq \{u\}$. We want to apply Theorem 8.27 on the formula $F = \mathbb{C}_v \cup \mathbb{B}(V_v)$, which as we recall implies $\mathit{All}^+(P^v)$ minimally. To this end, let $S' = S \setminus \{u\}$, write $S' = S_1 \,\dot{\cup}\, S_2$ for $S_1 = S' \cap V_v$ and $S_2 = S' \setminus S_1$, and consider

$$\begin{aligned} F_{S'} &= \{C \in (\mathbb{C}_v \cup \mathbb{B}(V_v)) \mid V(C) \cap S' \neq \emptyset\} \\ &= \mathbb{C}_v[\![S']\!] \cup \mathbb{B}(S_1) \ . \end{aligned} \tag{8.9}$$

For each $w \in S_1$, the clauses in $\mathbb{B}(S_1)$ contain d literals $w_1, \ldots, w_d$, and these literals must all occur negated in $\mathbb{C}_v$ by Lemma 8.26. For each $w \in S_2$, the clauses in $\mathbb{C}_v[\![S']\!]$ contain at least one variable w_i. Appealing to Theorem 8.27 with the subset of variables $\mathit{Vars}^d(S') \cap \mathit{Vars}(\mathbb{C}_v) \subseteq \mathit{Vars}(\mathbb{C}_v \cup \mathbb{B}(V_v)) \setminus \mathit{Vars}(\mathit{All}^+(P^v))$, we get

$$\begin{aligned}\left|F_{S'}\right| &= \left|\mathbb{C}_v[\![S']\!] \cup \mathbb{B}(S_1)\right| \\ &\geq \left| \mathit{Vars}^d(S') \cap \mathit{Vars}(\mathbb{C}_v)\right| + 1 \\ &\geq d\left|S_1\right| + \left|S_2\right| + 1 \ ,\end{aligned} \tag{8.10}$$

and rewriting this as

$$\begin{aligned}\left|\mathbb{C}_v[\![S]\!]\right| &\geq \left|\mathbb{C}_v[\![S']\!]\right| \\ &= \left|F_{S'}\right| - \left|\mathbb{B}(S_1)\right| \\ &\geq (d-1)\left|S_1\right| + \left|S_2\right| + 1 \\ &\geq \left|S\right|\end{aligned} \tag{8.11}$$

proves the claim (this is where we use that $d \geq 2$).

Note that $\mathbb{C}_v[\![S]\!] \subseteq \mathbb{C}[\![U]\!]$, since $\mathbb{C}_v \subseteq \mathbb{C}$ and $S \subseteq U$. Also, by construction $\mathbb{C}_v[\![S]\!]$ does not mention any vertices in $U \setminus S$ since $S = U \cap V(\mathbb{C}_v)$. In other words, $\mathbb{C}[\![U]\!] \supseteq \mathbb{C}_v[\![S]\!] \cup \mathbb{C}[\![U \setminus S]\!]$ for $\mathbb{C}_v[\![S]\!] \cap \mathbb{C}[\![U \setminus S]\!] = \emptyset$, and using the induction hypothesis for $U \setminus S \subsetneqq U$ we get

$$\left|\mathbb{C}[\![U]\!]\right| \geq \left|\mathbb{C}_v[\![S]\!]\right| + \left|\mathbb{C}[\![U \setminus S]\!]\right| \geq |S| + |U \setminus S| = |U|. \tag{8.12}$$

The theorem follows by induction. □

We can now prove Theorem 8.2 on page 98.

Theorem 8.2 (restated). *If π is a resolution refutation of a pebbling contradiction $\mathit{Peb}^d_{T_h}$ of degree $d > 1$ and $\mathcal{L}_\pi$ is the associated labelled pebbling from Theorem 8.1, then $\mathsf{cost}(\mathcal{L}_\pi) = \mathrm{O}(Sp(\pi))$.*

Proof. As in the proof of Theorem 8.1, given a refutation $\pi : \mathit{Peb}^d_{T_h} \vdash 0$, we use Lemma 8.9 to get a derivation $\pi^* = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ of $\bigvee_{i=1}^d z_i$ from $^*\mathit{Peb}^d_{T_h}$ in the same space and consider the L-pebbling $\mathcal{L}_\pi$ constructed in Theorem 8.22. Then

$$\mathsf{cost}(\mathcal{L}_\pi) = \mathrm{O}\big(\max_{t\in[\tau]}\{\mathsf{cost}(\mathbb{L}(\mathbb{C}_t))\}\big) \tag{8.13}$$

by Theorem 8.22, and for all $t \in [\tau]$ it holds that

$$\mathsf{cost}(\mathbb{L}(\mathbb{C}_t)) \leq |\mathbb{C}_t| \tag{8.14}$$

by Theorem 8.29. Thus

$$\mathsf{cost}(\mathcal{L}_\pi) = \mathrm{O}\big(\max_{t\in[\tau]}\{|\mathbb{C}_t|\}\big) = \mathrm{O}\big(Sp(\pi^*)\big) = \mathrm{O}\big(Sp(\pi)\big) \tag{8.15}$$

and the theorem follows. □

The proof of the tight bound for the refutation clause space of pebbling contradictions over binary trees in Theorem 2.1 as presented in Section 8.1 is thereby complete.

8.5 The Labelled Pebbling Price of Binary Trees

In this final section we present a proof of Theorem 8.7 on page 103, i.e., that for binary trees, the L-pebbling price coincides with the black-white pebbling price up to (small) constant factors. Since the argument is quite lengthy, we begin by giving an outline of its structure.

8.5.1 High-Level Overview of Proof

The proof of the theorem consists of two main components. The first component is pretty straightforward and is taken care of in Section 8.5.2. The second component is much more involved and takes up the rest of the section. In this first subsection we discuss these two parts of the proof informally, state the two corresponding formal lemmas that we will need, and show how they together yield Theorem 8.7.

For the first part, studying Definition 8.5 on page 101 carefully, one can argue that if we remove the reversal rule from the labelled pebble game, what remains looks essentially just like a disguised version of the standard black-white pebble game in Definition 5.1 on page 49. True, the rule for white pebble removal has been somewhat changed, and we are grouping pebbles together in pebble subconfigurations, but if we take any "sensible" L-pebbling $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$, ignore this pebble grouping, and just look at how the set of all black and white pebbles $\bigl(Bl(\mathbb{L}_t), Wh(\mathbb{L}_t)\bigr)$ changes over time with t, it seems plausible that we should obtain something pretty close to a standard black-white pebbling.

This is indeed the case, and we formalize the intuition above in Section 8.5.2, where we prove the following lemma.

Lemma 8.30. *Suppose that G is an arbitrary DAG with unique sink, and let $\mathcal{L}$ be any complete L-pebbling of G without reversal moves. Then from $\mathcal{L}$ we can construct a complete black-white pebbling $\mathcal{P}$ of G such that $\mathsf{cost}(\mathcal{P}) \leq \mathsf{cost}(\mathcal{L})$.*

Thus, if we could somehow do away with the reversal moves without increasing pebbling price, we would be done. Recall that we know from Lemma 8.6 on page 102 that for general DAGs, this *cannot* be done. The counterexample in Lemma 8.6 does not apply to binary trees, though. Rather on the contrary, toying around with L-pebblings of small binary trees, one cannot help getting the feeling that the removal of reversals should not affect the L-pebbling price in any way whatsoever. To show this formally, we need to make a detailed analysis of L-pebblings of trees and find out what structural properties can help us get rid of reversal moves in this special case. This is the second, much harder, component in the proof of Theorem 8.7.

In Section 8.5.3, we present some further definitions and notation that we will use when studying this problem, and make some useful technical observations. The rest of the section is then spent proving that for binary trees, the rule for reversal can in fact be omitted from the L-pebble game. We do not quite get the result that the pebbling price is not affected at all by this, but we show that it cannot increase by more than a constant factor 2. Such a bound is wholly sufficient for our purposes.

Unfortunately, the proof of this fact is *very* technical, but the structure of the underlying argument is not that complicated. Below, we try to sketch what it looks like to give the reader an idea of where we are going.

1. We first take care of a minor technical issue. In the pebble configurations of the standard black-white pebble game, we have black pebbles, and below each black pebble the white pebbles it depends on with nothing in between. In contrast, in the L-pebble game there can be other black pebbles in between a black pebble and its white pebbles, or two black pebbles one above the other without any white pebbles below them (see, for example, $v_1\langle v_2, v_6\rangle$ and $v_7\langle\emptyset\rangle$ in Figure 8.7 on page 129, or $v\langle v_1, v_2, v_3\rangle$ and $w\langle w_4, w_5\rangle$ in Figure 8.8(a) on page 134).

   The first step in our elimination of reversal moves is to show that this difference is inconsequential. Namely, we establish that without loss of generality we can assume that any L-pebbling $\mathcal{L}$ is *non-overlapping* in the sense that, roughly speaking, different pebble subconfigurations in the same labelled pebble configuration do not intersect (Definition 8.50 and Lemma 8.55 in Section 8.5.4).

2. Next, we study the connection between reversal moves and the set of vertices covered by the pebble subconfigurations in the sense of Definition 8.4 on page 100. In a standard black-white pebbling of a binary tree T, the set of vertices covered (generalizing Definition 8.4 in the natural way) expands monotonically as the pebbling proceeds, but in an L-pebbling it might also shrink as a result of reversal moves.

   As was discussed above, our intuition is that for trees this "shrinking" should not help to produce cheaper pebblings. As a part of our attempt to understand what happens during reversal moves, we observe that if we restrict an L-pebbling $\mathcal{L}$ to a subset of the vertices in T and let $\mathcal{L}$ act on these vertices in the natural way, we get a legal L-pebbling on this subset of vertices. We refer to this restriction operation as *projection* (Definition 8.51 and Lemma 8.57 in Section 8.5.5).

3. This leads to the idea of trying to get rid of reversal moves altogether in the following way: When the cover of a labelled pebble configuration shrinks as the result of a reversal move, we eliminate this reversal by projecting the L-pebbling moves made so far on what remains *after* the reversal move. We know that every such projection results in a legal L-pebbling, and if we do this by forward induction for all reversal moves in $\mathcal{L}$, we get a reversal-free complete L-pebbling $\mathcal{L}'$ of T (Section 8.5.6).

4. The problem is that these projection operations do not preserve pebbling cost—the pebbling $\mathcal{L}$ may contain reversal moves such that the projected

pebbling $\mathcal{L}'$ becomes more expensive than $\mathcal{L}$. We identify which kind of reversals in $\mathcal{L}$ spoil our construction of a reversal-free and cheap pebbling $\mathcal{L}'$ by projection and note that, from a global perspective, such *wasteful* reversal moves seem clearly non-optimal (Example 8.58).

Encouraged by this, and allowing some temporary wishful thinking, we then demonstrate that for all L-pebblings that contain reversal moves but avoid this special class of wasteful reversals, the projection construction sketched above works (Definition 8.59 and Lemma 8.61 in Section 8.5.7).

5. In this way, the whole problem finally boils down to whether wasteful reversals can be eliminated. In general, we cannot assume that an L-pebbling $\mathcal{L}$ does not make wasteful reversal moves, but we show that if $\mathcal{L}$ contains such moves, we can construct another L-pebbling $\mathcal{L}'$ in which these wasteful reversals are replaced by stronger, non-wasteful moves without increasing the total pebbling cost by more than a constant factor (Lemma 8.66 in Section 8.5.8).

Summing this up, we get the next lemma.

Lemma 8.31. *Suppose that $\mathcal{L}$ is a complete L-pebbling of a complete binary tree T. Then from $\mathcal{L}$ we can construct a complete L-pebbling $\mathcal{L}'$ of T without reversals such that $\mathsf{cost}(\mathcal{L}') = \mathrm{O}(\mathsf{cost}(\mathcal{L}))$.*

Assuming Lemmas 8.30 and 8.31, it is easy to prove that the L-pebbling price and the black-white pebbling price of a complete binary tree T_h of height h coincide asymptotically.

Theorem 8.7 (restated). $\mathsf{L\text{-}Peb}(T_h) = \Theta\bigl(\mathsf{BW\text{-}Peb}(T_h)\bigr)$.

Proof. The black pebbling price of T_h is $\mathsf{Peb}(T_h) = \mathrm{O}(h) = \mathrm{O}(\mathsf{BW\text{-}Peb}(T_h))$ according to Theorem 5.2 on page 52. It is not hard to see that an L-pebbling $\mathcal{L}$ can imitate a black pebbling $\mathcal{P}$ in the same cost. For suppose that at some point in time t a black pebble is placed on the vertex r in $\mathcal{P}$. If r is a source, $\mathcal{L}$ can match this move by introducing $r\langle\emptyset\rangle$. Otherwise, if $\mathit{pred}(r) = \{p, q\}$, both these vertices must be black-pebbled at time t in $\mathcal{P}$, so by induction we have $p\langle\emptyset\rangle$ and $q\langle\emptyset\rangle$ in $\mathcal{L}$. Introducing $r\langle p, q\rangle$, merging with $p\langle\emptyset\rangle$ and $q\langle\emptyset\rangle$ on p and q, respectively, and then erasing $r\langle p, q\rangle$, we get $r\langle\emptyset\rangle$. Thus $\mathsf{L\text{-}Peb}(T_h) \leq \mathsf{Peb}(T_h) = \mathrm{O}(\mathsf{BW\text{-}Peb}(T_h))$.

In the other direction, let $\mathcal{L}$ be a complete L-pebbling of T_h in minimal cost. By Lemma 8.31, there exists a complete L-pebbling $\mathcal{L}'$ of T_h without reversal moves such that $\mathsf{cost}(\mathcal{L}') = \mathrm{O}(\mathsf{cost}(\mathcal{L}))$. By Lemma 8.30 we can construct a plain old black-white pebbling $\mathcal{P}$ of T_h from $\mathcal{L}'$ for which $\mathsf{cost}(\mathcal{P}) \leq \mathsf{cost}(\mathcal{L}')$. Hence $\mathsf{BW\text{-}Peb}(T_h) = \mathrm{O}(\mathsf{L\text{-}Peb}(T_h))$, and the theorem follows. □

So all that needs to be done is to prove Lemmas 8.30 and 8.31, which we do starting in the next subsection.

We make one final remark before plunging into the proofs. We are aware that the technical machinery in this section can appear cumbersome. However, this might mainly be due to the fact that sometimes, one picture says more than the thousand words used to formalize it mathematically. We feel that at times in this section, we are forced to go to great lengths to prove statements that seem intuitively very plausible once one visualizes what they actually say. Therefore, we believe that the arguments should be possible to follow more easily if the reader tries to digest what the definitions mean and what is proven about them simply by drawing a binary tree of suitable height and working out small examples in this binary tree while reading.

8.5.2 Reversal-Free L-pebblings Are (Almost) Standard Pebblings

We present the proof of Lemma 8.30 in two steps, one easy and one harder.

The first modification of the pebble game when going from Definition 5.1 to Definition 8.5 was that in the context of resolution, it appears that a more natural rule for white pebble removal is that a white pebble can be removed from a vertex when a black pebble is placed on that same vertex. It is thanks to this that we get the close correspondence between clauses and pebbles in Section 8.3.

It seems intuitively fairly obvious that this rule change should not really affect the pebble game, but for completeness we state and prove this fact formally.

Definition 8.32 (S-pebble game). Suppose that G is a DAG with unique sink z. The *superpositioned black-white pebble game*, or *S-pebble game*, is as in Definition 5.1, except that a vertex may have both a black and a white pebble on itself, and the pebbling rules are (1)–(3) in Definition 5.1 and (4') below instead of rule (4) in Definition 5.1.

4'. A white pebble on v can be removed only if there is a black pebble on v.

We write $\mathsf{S\text{-}Peb}(G)$ to denote the minimum cost of any complete S-pebbling of G.

Lemma 8.33. *For any DAG G it holds that $\mathsf{S\text{-}Peb}(G) = \mathsf{BW\text{-}Peb}(G)$.*

Proof. It is easy to see that for any standard black-white pebbling $\mathcal{P}$ of G we can make an S-pebbling $\mathcal{S}$ of G in exactly the same cost. Every white pebble removal from a vertex v in $\mathcal{P}$ according to rule (4) corresponds to first placing a black pebble on v in $\mathcal{S}$ in no extra cost and then removing first the white pebble according to rule (4') and then the black pebble according to rule (2).

In the other direction, suppose that we are given a superpositioned pebbling $\mathcal{S} = \{\mathbb{S}_0, \ldots, \mathbb{S}_\tau\}$ of G. We construct a standard black-white pebbling $\mathcal{P} = \{\mathbb{P}_0, \ldots, \mathbb{P}_\tau\}$ such that for $\mathbb{P}_t = (B_t, W_t)$ and $\mathbb{S}_t = (B'_t, W'_t)$ it holds that $B_t = B'_t$, $B_t \cup W_t = B'_t \cup W'_t$, and (as required by Definition 5.1) $B_t \cap W_t = \emptyset$. In particular, this means that $\mathsf{cost}(\mathcal{P}) = \mathsf{cost}(\mathcal{S})$ and that if $\mathcal{S}$ is a complete pebbling, then so is $\mathcal{P}$.

The construction is by forward induction over $\mathcal{S}$. We set $\mathbb{P}_0 = \mathbb{S}_0 = (\emptyset, \emptyset)$ and then make the inductive step by a case analysis over the pebbling moves.

1. If $\mathcal{S}$ places a black pebble on v at time $t+1$, the vertices in *pred*(v) must be pebbled in $\mathbb{S}_t$ and thus in $\mathbb{P}_t$. If $v \in W_t$, we remove the white pebble from v in $\mathcal{P}$. Then we place a black pebble on v.

2. If $\mathcal{S}$ removes a black pebble from v at time $t+1$, by induction v is black-pebbled and the vertices in *pred*(v) are pebbled in $\mathcal{P}$. Thus we can remove the black pebble from v in $\mathcal{P}$, and in case $v \in W'_t$ we then place a white pebble on v.

3. If $\mathcal{S}$ places a white pebble on v at time $t+1$, we place a white pebble there in $\mathcal{P}$ if $v \notin B_t$ and otherwise do nothing.

4. When a white pebble is removed from v in $\mathcal{S}$ it holds that $v \in B'_t$. Then by induction $v \in B_t$, so the white pebble has already been removed from v in $\mathcal{P}$ or was never placed there.

Note that to avoid being overly formalistic, we ignore the fact there there might be "idle moves" $\mathbb{P}_t = \mathbb{P}_{t+1}$ and moves simultaneously removing and placing a pebble on the same vertex in $\mathcal{P}$ and $\mathcal{S}$. It should be clear that this is not a problem. □

The second step in the proof of Lemma 8.30 is to show that if we take a complete L-pebbling $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$ of a DAG G without reversal moves and look at $\big(Bl(\mathbb{L}_t), Wh(\mathbb{L}_t)\big)$ for $t \in [\tau]$, we can extract a legal complete S-pebbling of G in at most the same cost. We prove this in the next two lemmas.

The first lemma says that without loss of generality we can assume that all L-pebblings are *non-redundant* in the sense that if a subconfiguration $v\langle V\rangle$ is derived at time t, then this subconfiguration is not just thrown away but is used at some time $t' > t$ further on in the pebbling before being erased.

From now on, in order not to clutter the notation we allow a mild abuse of notation by omitting curly brackets around singleton L-configurations, quite often writing, for instance, $v\langle V\rangle \preceq \mathbb{L}$, $u\langle U\rangle = \mathbb{L}$, and $\mathbb{L} \cup w\langle W\rangle$ instead of $\{v\langle V\rangle\} \preceq \mathbb{L}$, $\{u\langle U\rangle\} = \mathbb{L}$, and $\mathbb{L} \cup \{w\langle W\rangle\}$. Also, we sometimes drop the curly brackets around singleton sets within subconfigurations, writing, for instance, $v\langle (V \cup W) \setminus w\rangle$ instead of $v\langle (V \cup W) \setminus \{w\}\rangle$ for the merger of $v\langle V\rangle$ and $w\langle W\rangle$.

Lemma 8.34. *Let $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$ be an arbitrary complete L-pebbling of a DAG G. Then we can construct a complete L-pebbling $\mathcal{L}' = \{\mathbb{L}'_0, \ldots, \mathbb{L}'_{\tau'}\}$ of G with $\mathsf{cost}(\mathcal{L}') \leq \mathsf{cost}(\mathcal{L})$ that has the following property: If $v\langle V\rangle$ is erased at time t in $\mathcal{L}'$, i.e., $v\langle V\rangle \in \mathbb{L}'_t \setminus \mathbb{L}'_{t+1}$, then this subconfiguration has been used in a merger or reversal move immediately before being erased, and the subconfiguration resulting from this move is present in $\mathbb{L}'_{t+1}$. Also, if $\mathcal{L}$ makes no reversal moves, then neither does $\mathcal{L}'$.*

Proof. Let us first try to visualize the proof. For any L-pebbling $\mathcal{L}$, we can construct a DAG $G_{\mathcal{L}}$ encoding the pebbling as follows. For every subconfiguration $v\langle V\rangle$ appearing at time t_1 and staying in the graph until time t_2 when it is erased, we create a vertex $(v\langle V\rangle, [t_1, t_2])$. For each reversal $u\langle U\rangle \rightsquigarrow v\langle V\rangle$, we draw an edge from the vertex representing this occurrence of $u\langle U\rangle$ to the vertex representing this occurrence of $v\langle V\rangle$. For each merger $u\langle U\rangle = \mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$, we draw edges from $v\langle V\rangle$ and $w\langle W\rangle$ to $u\langle U\rangle$. The sources in $G_{\mathcal{L}}$ are vertices $(v\langle \mathit{pred}(v)\rangle, [t_1, t_2])$, and by assumption there is a sink $(z\langle\emptyset\rangle, [t_1, \tau])$. Note that by the definition of the L-pebble game we never derive a subconfiguration that is already present in the graph, so all vertices in $G_{\mathcal{L}}$ have indegree 0, 1 or 2 corresponding to introductions, reversals and mergers.

Consider the subgraph of $G_{\mathcal{L}}$ consisting of all vertices from which the sink vertex $(z\langle\emptyset\rangle, [t_1, \tau])$ is reachable. We construct $\mathcal{L}'$ to be the subpebbling corresponding exactly to the moves in this subgraph, except that erasures are always performed as soon as possible. Since the moves in $\mathcal{L}'$ are a subset of the moves in $\mathcal{L}$, clearly $\mathcal{L}'$ is reversal-free if $\mathcal{L}$ is.

Formally, this amounts to the following. We construct the modified pebbling $\mathcal{L}'$ by backward induction over $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$. Let $\mathbb{L}'_\tau = \mathbb{L}_\tau = \{z\langle\emptyset\rangle\}$. Our induction hypothesis is that $\mathbb{L}'_{t^*} \subseteq \mathbb{L}_{t^*}$ for $t^* > t$. The backward induction step from $t+1$ to t is a case analysis over the moves $\mathbb{L}_t \rightsquigarrow \mathbb{L}_{t+1}$ in $\mathcal{L}$. For simplicity, we allow using fractional time steps in the interval $[t, t+1]$ in the inductive constructions below.

Introduction $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle \mathit{pred}(v)\rangle$: Set $\mathbb{L}'_t = \mathbb{L}'_{t+1} \setminus v\langle \mathit{pred}(v)\rangle$. Note that we might have $\mathbb{L}'_t = \mathbb{L}'_{t+1}$ if $v\langle \mathit{pred}(v)\rangle \notin \mathbb{L}'_{t+1}$. In any case, the induction hypothesis holds for $\mathbb{L}'_t$.

Merger $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle (V \cup W)\setminus w\rangle$: If $v\langle (V \cup W)\setminus w\rangle \notin \mathbb{L}'_{t+1}$, set $\mathbb{L}'_t = \mathbb{L}'_{t+1}$. The induction hypothesis trivially remains true. Otherwise, if the merged subconfiguration is present in $\mathbb{L}'_{t+1}$, set $\mathbb{L}'_t = \bigl(\mathbb{L}'_{t+1} \cup \{v\langle V\rangle, w\langle W\rangle\}\bigr)\setminus v\langle (V \cup W)\setminus w\rangle$. We can go from $\mathbb{L}'_t$ to $\mathbb{L}'_{t+1}$ in at most three steps via intermediate L-configurations $\mathbb{L}'_{t+1/3} = \mathbb{L}'_t \cup v\langle (V \cup W) \setminus w\rangle$ and $\mathbb{L}'_{t+2/3} = \mathbb{L}'_{t+1} \cup w\langle W\rangle$ by first merging $v\langle V\rangle$ and $w\langle W\rangle$, then possibly erasing $v\langle V\rangle$, and finally possibly erasing $w\langle W\rangle$.

Reversal $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle V\rangle$ for $v\langle V\rangle \prec u\langle U\rangle \in \mathbb{L}_t$: If $v\langle V\rangle \notin \mathbb{L}'_{t+1}$, set $\mathbb{L}'_t = \mathbb{L}'_{t+1}$. Otherwise, set $\mathbb{L}'_t = \bigl(\mathbb{L}'_{t+1} \cup u\langle U\rangle\bigr)\setminus v\langle V\rangle$. We can go from $\mathbb{L}'_t$ to $\mathbb{L}'_{t+1}$ in at most two steps via the intermediate L-configuration $\mathbb{L}'_{t+1/2} = \mathbb{L}'_{t+1} \cup u\langle U\rangle$, i.e., by first reversing $u\langle U\rangle$ to $v\langle V\rangle$ and then possibly erasing $u\langle U\rangle$.

Erasure $\mathbb{L}_{t+1} = \mathbb{L}_t \setminus v\langle V\rangle$: All erasure moves in $\mathcal{L}'$ are taken care of in connection with mergers or reversals, so set $\mathbb{L}'_t = \mathbb{L}'_{t+1}$.

We claim that all moves in $\mathcal{L}'$ constructed in this way are legal (if we eliminate repeated L-configurations $\mathbb{L}'_t = \mathbb{L}'_{t+1}$). For if $u\langle U\rangle \in \mathbb{L}'_t$, then $u\langle U\rangle \in \mathbb{L}_t$, and we know that this subconfiguration must have been derived at some point in time $t^* \leq t$ in $\mathcal{L}$ by introduction, merger, or reversal. Thus the backward construction of $\mathcal{L}'$ will yield a correct derivation of $u\langle U\rangle$. Also note that by the construction for the merger and reversal moves, when a subconfiguration in $\mathcal{L}'$ is erased it has just been used in some merger or reversal move.

Finally, by construction $\mathbb{L}'_t \subseteq \mathbb{L}_t$, and for the intermediate fractional time step L-configurations $\mathbb{L}'_{t+a/b}$ in the merger and reversal moves in $\mathcal{L}'$ we have $\mathbb{L}'_{t+a/b} \subseteq \mathbb{L}_{t+1}$. This shows that for all $\mathbb{L}' \in \mathcal{L}'$ there is a corresponding $\mathbb{L} \in \mathcal{L}$ such that $\mathsf{cost}(\mathbb{L}') \leq \mathsf{cost}(\mathbb{L})$ (part 1 of Proposition 8.21). It follows that $\mathsf{cost}(\mathcal{L}') \leq \mathsf{cost}(\mathcal{L})$. □

For L-pebblings as in Lemma 8.34, if we ignore all relations between black and white pebbles in the subconfigurations and consider $\big(Bl(\mathbb{L}_t), Wh(\mathbb{L}_t)\big)$ for $t \in [\tau]$, this is a legal S-pebbling.

Lemma 8.35. *Suppose that $\mathcal{L}$ is a complete L-pebbling of a DAG G without reversal moves. Then there is a complete S-pebbling $\mathcal{S}$ of G such that $\mathsf{cost}(\mathcal{S}) \leq \mathsf{cost}(\mathcal{L})$.*

Proof. By Lemma 8.34, without loss of generality we can assume that each $v\langle V\rangle$ is erased from $\mathcal{L}$ precisely after it has been used in a merger, and that $v\langle V\rangle$ is erased before $w\langle W\rangle$ when both subconfigurations are eliminated after a move $v\langle (V \cup W) \setminus w\rangle = \mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$, so that the white pebble on w is removed before the black pebble on w.

It is clear that we are done if we can construct an S-pebbling $\mathcal{S}$ with moves matching the moves in $\mathcal{L}$ exactly. Let $\mathbb{S}_0 = (\emptyset, \emptyset)$ and construct $\mathbb{S}_{t+1}$ inductively by looking at the moves in $\mathbb{L}_t \rightsquigarrow \mathbb{L}_{t+1}$.

Introduction $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle pred(v)\rangle$: Place white pebbles on $pred(v)$ and then a black pebble on v in $\mathcal{S}$.

Merger $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle (V \cup W) \setminus w\rangle$ for $v\langle V\rangle, w\langle W\rangle \in \mathbb{L}_t$: No pebbling moves in $\mathcal{S}$, but note that if $v\langle V\rangle$ is now removed, the change in pebbles on G in $\mathcal{L}$ is exactly the same as after an application of rule (4') on w.

Erasure $\mathbb{L}_{t+1} = \mathbb{L}_t \setminus v\langle V\rangle$: This is the only nontrivial case. In general, an erasure move in an L-pebbling can remove an arbitrary number of white pebbles without any black pebbles being even close to these white pebbles, and there is no way we can match such a move in an S-pebbling. But since we can assume that $\mathcal{L}$ is an L-pebbling as described in Lemma 8.34, we know that $v\langle V\rangle$ has just been used in a merger. Consequently, the only pebble that disappears when going from $\big(Bl(\mathbb{L}_t), Wh(\mathbb{L}_t)\big)$ to $\big(Bl(\mathbb{L}_{t+1}), Wh(\mathbb{L}_{t+1})\big)$ is either the black pebble on v, which is always a legal pebble removal, or some white pebble on $w \in V$ which has just been eliminated in the merger move by a black pebble, and this is a legal pebble removal according to rule (4').

We see that $\mathcal{S}$ generated in this way is a legal S-pebbling if we modify each introduction step into three pebble placement moves. Clearly, $\mathsf{cost}(\mathcal{S}) \leq \mathsf{cost}(\mathcal{L})$. The lemma follows. □

Combining Lemmas 8.33 and 8.35 immediately yields Lemma 8.30.

Lemma 8.30 (restated). *Suppose that G is an arbitrary DAG with unique sink, and let $\mathcal{L}$ be any complete L-pebbling of G without reversals. Then from $\mathcal{L}$ we can construct a complete black-white pebbling $\mathcal{P}$ of G such that $\mathsf{cost}(\mathcal{P}) \leq \mathsf{cost}(\mathcal{L})$.*

Proof. Given any L-pebbling $\mathcal{L}$ of G without reversal moves, we use Lemma 8.35 to find an S-pebbling $\mathcal{S}$ in at most the same cost as $\mathcal{L}$. Then Lemma 8.33 helps us to transform $\mathcal{S}$ to a standard black-white pebbling $\mathcal{P}$ in at most the same cost as $\mathcal{S}$. □

8.5.3 Some Technical Preliminaries

In the rest of this section, we restrict our attention to binary trees and show that for such graphs the reversal rule can be omitted in the labelled pebble game. Before beginning to construct the proof of this statement, in this subsection we collect a number of technical observations that will simplify matters later on. In the process, we also introduce some more definitions and notation.

Recall the terminology and notation from the beginning of Section 8.2 and from Definitions 8.10 and 8.11. We add that, in this section, P and Q will denote paths in T. Also, if $succ(u) = succ(v)$ for $u \neq v$, we will say that u and v are *siblings* and write $v = sibl(u)$. Note that siblings are unrelated vertices in the sense of Section 8.2; i.e., there is no (directed) path between u and v.

We observe that for binary trees, the cover of a subconfiguration can be defined more explicitly than in Definition 8.4, and also has the following convexity property.

Definition 8.36. We say that a vertex set $V \subseteq V(G)$ in a DAG G is *convex* if for all $u_1, u_2 \in V$ there is a $u^* \in V$ above both u_1 and u_2 such that for all paths $P_i : u_i \rightsquigarrow u^*$, $i = 1, 2$, it holds that $P_i \subseteq V$.

Proposition 8.37. *For any pebble subconfiguration $v\langle W\rangle$ in a binary tree T it holds that $cover(v\langle W\rangle) = T^v \setminus \bigcup_{w \in W} T^w$. In particular, $cover(v\langle W\rangle)$ is a convex set.*

This is not true for general DAGs. Consider, for instance, the pyramid graph of height 4 with vertex labels as in Figure 8.3 on page 103. Then for $z\langle u_2, u_3\rangle$ it holds that $s_2, s_4 \in cover(z\langle u_2, u_3\rangle)$, but for any vertex above both s_2 and s_4 we can always pick paths going through $u_2, u_3 \notin cover(z\langle u_2, u_3\rangle)$ so $cover(z\langle u_2, u_3\rangle)$ is not convex.

Proof of Proposition 8.37. Proving the set inclusion $T^v \setminus \bigcup_{w \in W} T^w \subseteq cover(v\langle W\rangle)$ is straightforward. Since all vertices $u \in T^v \setminus \bigcup_{w \in W} T^w$ are below v, they have paths $P : u \rightsquigarrow v$ to v. But no u is below any $w \in W$, so the paths P cannot possibly intersect W. Thus $u \in cover(v\langle W\rangle)$ according to Definition 8.4.

To show that this is an equality, we have to make use of the fact that T is a tree. Namely, this implies that the path $P : u \rightsquigarrow v$, if it exists, must be unique. Suppose to get a contradiction that $u \in cover(v\langle W\rangle)$ but $u \notin T^v \setminus \bigcup_{w \in W} T^w$. By definition there is a path $P : u \rightsquigarrow v$, so $u \in T^v$. It follows that there must exist some $w \in W$ such that $u \in T^w$. But then the unique path $P : u \rightsquigarrow v$, must pass through w, so $P \cap W \neq \emptyset$, contradicting the assumption that $u \in \mathit{cover}(v\langle W\rangle)$.

To prove convexity, just set $u^* = v$ in Definition 8.36 and use that the path between any two vertices in T is uniquely determined. □

A nice property of mergers in binary trees is that if we merge two simple subconfigurations (Definition 8.11), then the resulting subconfiguration is also simple. We remark that this is not true in more general DAGs. If we look at Figure 8.3 again, the subconfigurations $z\langle x_2, u_2, u_3\rangle$ and $x_2\langle s_3\rangle$ are both simple, but their merger $z\langle u_2, u_3, s_3\rangle$ is not.

Observation 8.38. *If $v\langle V\rangle$ and $w\langle W\rangle$ with $w \in V$ are simple subconfigurations in a binary tree, then $\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$ is also simple.*

Proof. By definition, $\mathsf{merge}(v\langle V\rangle, w\langle W\rangle) = v\langle (V \cup W) \setminus \{w\}\rangle$. Since V is simple and we are in a binary tree, it holds that $T^w \cap \bigcup_{x \in V\setminus\{w\}} T^x = \emptyset$. To get the required paths from $u \in W$ to v in Definition 8.11, just concatenate the paths from $u \in W$ to w with the path from w to v. □

Another nice property of mergers of simple subconfigurations is that the cover of a merger is the disjoint union $\dot\cup$ of the covers of the merged subconfigurations. Figure 8.2 on page 102 provides an illustration of this. Again, this holds only in the binary tree case. Reusing the example subconfigurations $z\langle x_2, u_2, u_3\rangle$ and $x_2\langle s_3\rangle$ above, it is readily verified that we have $\mathit{cover}(\mathsf{merge}(z\langle x_2, u_2, u_3\rangle, x_2\langle s_3\rangle)) = \mathit{cover}(z\langle u_2, u_3, s_3\rangle) \neq \mathit{cover}(z\langle x_2, u_2, u_3\rangle) \cup \mathit{cover}(x_2\langle s_3\rangle)$.

Proposition 8.39. *Suppose that $u\langle U\rangle$, $v\langle V\rangle$, and $w\langle W\rangle$ are simple pebble subconfigurations in a binary tree. Then it holds that $u\langle U\rangle = \mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$ if and only if $\mathit{cover}(u\langle U\rangle) = \mathit{cover}(v\langle V\rangle) \mathbin{\dot\cup} \mathit{cover}(w\langle W\rangle)$.*

Proof. ($\Rightarrow$) If $u\langle U\rangle = \mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$, it holds that $w \in V$, and since we are in a tree and V is a simple set, we have $T^w \cap \bigcup_{x \in V\setminus w} T^x = \emptyset$. Combining this with the fact that W is below w by definition, we get $\bigcup_{y \in W} T^y \subseteq T^w$ and $\bigcup_{x \in V\setminus w} T^x \cap \bigcup_{y \in W} T^y = \emptyset$. The equality in the proposition follows by using Proposition 8.37 and checking that

$$\begin{aligned} \mathit{cover}(u\langle U\rangle) &= \mathit{cover}\bigl(v\bigl\langle (V \cup W) \setminus w\bigr\rangle\bigr) \\ &= T^v \setminus \bigcup_{x \in (V\cup W)\setminus w} T^x \\ &= T^v \setminus \left(\bigcup_{x \in V\setminus w} T^x \mathbin{\dot\cup} \bigcup_{y \in W} T^y \right) \\ &= \left(T^v \setminus \bigcup_{x \in V} T^x \right) \mathbin{\dot\cup} \left(T^w \setminus \bigcup_{y \in W} T^y \right) \\ &= \mathit{cover}(v\langle V\rangle) \mathbin{\dot\cup} \mathit{cover}(w\langle W\rangle) \ . \end{aligned} \tag{8.16}$$

($\Leftarrow$) Suppose that $\mathit{cover}(u\langle U\rangle) = \mathit{cover}(v\langle V\rangle) \mathbin{\dot\cup} \mathit{cover}(w\langle W\rangle)$. Since by definition all vertices in $\mathit{cover}(v\langle V\rangle)$ and $\mathit{cover}(w\langle W\rangle)$ are below v and w, respectively, but $\mathit{cover}(v\langle V\rangle) \cup \mathit{cover}(w\langle W\rangle) = \mathit{cover}(u\langle U\rangle)$ is convex by Proposition 8.37, setting $u_1 = v$ and $u_2 = w$ in Definition 8.36 shows that either v is below w or w is below v. Suppose without loss of generality that the latter case holds. Then using the same reasoning again we see that $v = u$.

Since $w \in \mathit{cover}(u\langle U\rangle)$ there is a path $P : w \rightsquigarrow u$ with $P \subseteq \mathit{cover}(u\langle U\rangle)$. Clearly, $(P \setminus w) \cap \mathit{cover}(w\langle W\rangle) = \emptyset$. Because $\mathit{cover}(v\langle V\rangle) \cap \mathit{cover}(w\langle W\rangle) = \emptyset$ by assumption and $w \in \mathit{cover}(w\langle W\rangle)$ by definition, we must have $(P \setminus w) \subseteq \mathit{cover}(v\langle V\rangle)$ but $w \notin \mathit{cover}(v\langle V\rangle)$. Applying Proposition 8.37 we see that $w \in V$, so $v\langle V\rangle$ and $w\langle W\rangle$ are mergeable. By assumption, $u\langle U\rangle$, $v\langle V\rangle$, and $w\langle W\rangle$ are all simple subconfigurations, and using Proposition 8.37 again as well as the $\Rightarrow$-direction of this proposition it can be verified that the equality

$$\begin{aligned} \mathit{cover}(u\langle U\rangle) = T^u \setminus \bigcup_{x \in U} T^x = \mathit{cover}(v\langle V\rangle) \mathbin{\dot\cup} \mathit{cover}(w\langle W\rangle) = \\ = \mathit{cover}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) = T^v \setminus \bigcup_{x \in (V \cup W) \setminus w} T^x \qquad (8.17) \end{aligned}$$

can hold only if $U = (V \cup W) \setminus w$, i.e., only if $u\langle U\rangle = \mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$. □

Observe that we need the simplicity of the subconfigurations in order for Proposition 8.39 to hold. If $v\langle V\rangle$ were not simple, white pebbles in $\bigcup_{x \in V} T^x_*$ would create problems. In a sense, requiring that subconfigurations be simple is a way of ensuring that mergers behave in the way one would expect them to.

Now the subconfigurations $v\langle \mathit{pred}(v)\rangle$ in introduction moves are obviously simple, and Observation 8.38 says that mergers preserve simplicity. It is not hard to show that we can also assume that reversal moves result in simple subconfigurations, so that in any L-pebbling of a binary tree T it is always the case that all subconfigurations are simple. We next sketch a proof of this statement,[2] which will simplify matters in what follows.

Lemma 8.40. *Suppose that $\mathcal{L}$ is a complete L-pebbling of a binary tree T. Then from $\mathcal{L}$ we can construct a complete L-pebbling $\mathcal{L}'$ such that $\mathsf{cost}(\mathcal{L}') \leq \mathsf{cost}(\mathcal{L})$ and $\mathcal{L}'$ contains only simple L-configurations.*

Proof sketch. Recalling Definition 8.12, let $\mathbb{L}'_t = \{v\langle \mathit{swp}(v, W)\rangle \mid v\langle W\rangle \in \mathbb{L}_t\}$ for $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$. This implies that $\mathit{cover}(\mathbb{L}'_t) = \mathit{cover}(\mathbb{L}_t)$ (perhaps most easily seen by using Proposition 8.37) for $\mathbb{L}'_t$ consisting of simple subconfigurations. We

[2] However, we note that the reader who so wishes can instead make Lemma 8.40 an assumption and restrict Theorem 8.7 to the case of L-pebblings with simple subconfigurations. This is so since a careful reading of Section 8.3 reveals that the L-pebblings that we get from resolution derivations satisfy this property. To see this, note that by Definition 8.13 all subconfigurations in $\mathbb{L}(\mathbb{C}_t)$ are simple, and by Observation 8.38 and the construction in Lemma 8.20 all subconfigurations in the intermediate L-configurations are simple as well.

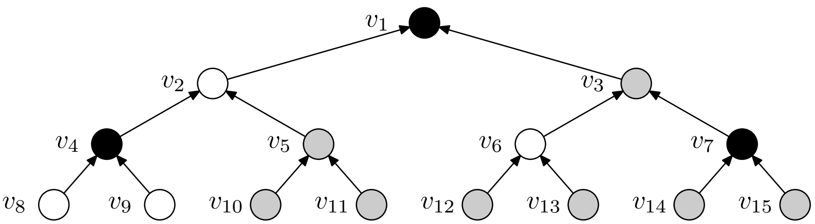


Figure 8.7: Three pebble subconfigurations $v_1\langle v_2, v_6\rangle$, $v_4\langle v_8, v_9\rangle$, and $v_7\langle\emptyset\rangle$.

claim that $\mathcal{L}' = \{\mathbb{L}'_0, \ldots, \mathbb{L}'_\tau\}$ is a legal L-pebbling if repeated L-configurations $\mathbb{L}'_t = \mathbb{L}'_{t+1}$ are eliminated. Let us outline the proof.

Introduction moves in $\mathcal{L}$ are always performed also in $\mathcal{L}'$, since $v\langle \mathit{pred}(v)\rangle = v\langle \mathit{swp}(v, \mathit{pred}(v))\rangle$.

Suppose that $v_1\langle W_1\rangle$ and $v_2\langle W_2\rangle$ are merged in $\mathcal{L}$, and let $W'_i = \mathit{swp}(v_i, W_i)$ for $i = 1, 2$. If $v_2 \notin W'_1$ we have $\mathit{swp}(v_1, (W_1 \cup W_2) \setminus v_2) = W'_1$, so nothing happens in $\mathcal{L}'$. Otherwise we can merge $v_1\langle W'_1\rangle$ and $v_2\langle W'_2\rangle$, and it is straightforward to verify that $\mathit{swp}(v_1, (W_1 \cup W_2) \setminus v_2) = \mathit{swp}(v_1, (W'_1 \cup W'_2) \setminus v_2)$.

Likewise, if $v_1\langle W_1\rangle$ is reversed to $v_2\langle W_2\rangle$ in $\mathcal{L}$, going from $v_1\langle W'_1\rangle$ to $v_2\langle W'_2\rangle$ in $\mathcal{L}'$ is a legal reversal move.

Finally, note that erasures are taken care of automatically by the definition of $\mathbb{L}'_t$. □

In the outline of the proof in Section 8.5.1, we said that we wanted to construct L-pebblings with "non-intersecting" subconfigurations. We next formally define two slightly different flavours of "non-intersecting" that we will use extensively below. It might be easier to parse this rather technical definition by first studying Examples 8.42 and 8.43.

Definition 8.41. For a simple pebble subconfiguration $v\langle W\rangle$, we define the *boundary* of $v\langle W\rangle$ to be $\partial v\langle W\rangle = \{v\} \cup W$. The *interior* of $v\langle W\rangle$ is $\mathit{int}(v\langle W\rangle) = \mathit{cover}(v\langle W\rangle) \setminus \partial v\langle W\rangle$ and the *closure* is $\mathit{cl}(v\langle W\rangle) = \mathit{cover}(v\langle W\rangle) \cup \partial v\langle W\rangle$.

If $\mathit{cover}(v\langle V\rangle) \cap \mathit{cover}(u\langle U\rangle) = \emptyset$, the subconfigurations $v\langle V\rangle$ and $u\langle U\rangle$ are said to be *non-overlapping*. If $\mathit{cl}(v\langle V\rangle) \cap \mathit{cl}(u\langle U\rangle) = \emptyset$, $v\langle V\rangle$ and $u\langle U\rangle$ are *non-touching*.

Example 8.42. Consider the subconfigurations in Figure 8.7 (which is Figure 8.1 but with all vertices labelled). For $v_1\langle v_2, v_6\rangle$ we have

$$\begin{aligned}
\mathit{cover}(v_1\langle v_2, v_6\rangle) &= \{v_1, v_3, v_7, v_{14}, v_{15}\}, \\
\partial v_1\langle v_2, v_6\rangle &= \{v_1, v_2, v_6\}, \\
\mathit{int}(v_1\langle v_2, v_6\rangle) &= \{v_3, v_7, v_{14}, v_{15}\}, \\
\mathit{cl}(v_1\langle v_2, v_6\rangle) &= \{v_1, v_2, v_3, v_6, v_7, v_{14}, v_{15}\}.
\end{aligned}$$

Since $cl(v_4\langle v_8, v_9\rangle) = \{v_4, v_8, v_9\}$, the subconfigurations $v_1\langle v_2, v_6\rangle$ and $v_4\langle v_8, v_9\rangle$ are non-touching. For $v_7\langle\emptyset\rangle$ we have $\mathit{cover}(v_7\langle\emptyset\rangle) = \{v_7, v_{14}, v_{15}\}$, so $v_7\langle\emptyset\rangle$ and $v_1\langle v_2, v_6\rangle$ are overlapping, or, more precisely, it holds that $v_7\langle\emptyset\rangle \prec v_1\langle v_2, v_6\rangle$.

Example 8.43. More generally, if $v\langle V\rangle$ and $w\langle W\rangle$ are simple, mergeable subconfigurations with $w \in V$, then $v\langle V\rangle$ and $w\langle W\rangle$ are non-overlapping (because of Proposition 8.39) but touching in w. This is illustrated in Figure 8.2.

For the case of binary trees, it turns out that Lemma 8.34 can be formulated more sharply. Remember that the cover of an L-configuration $\mathbb{L}$ is defined by taking the union of the covers of the subconfigurations in $\mathbb{L}$.

Lemma 8.44. *Suppose that $\mathcal{L} = \{\mathbb{L}_0, \dots, \mathbb{L}_\tau\}$ is a reversal-free L-pebbling on T such that all L-configurations $\mathbb{L}_t$ are simple and $\mathbb{L}_\tau$ consisting of pairwise non-overlapping subconfigurations. Then there is a reversal-free complete L-pebbling $\mathcal{L}' = \{\mathbb{L}'_0, \dots, \mathbb{L}'_{\tau'}\}$ with $\mathbb{L}'_0 \subseteq \mathbb{L}_0$, $\mathbb{L}'_{\tau'} = \mathbb{L}_\tau$ and $\mathsf{cost}(\mathcal{L}') \leq \mathsf{cost}(\mathcal{L})$ such that every $v\langle V\rangle$ in $\mathcal{L}'$ occurs during one contiguous time interval, and every $v\langle V\rangle$ in $\mathcal{L}'$ except those in $\mathbb{L}_\tau$ is used in exactly one merger, after which it is erased. Also, all $\mathbb{L}'_t$ are simple, and $\mathit{cover}(\mathbb{L}'_t)$ grows monotonically with t.*

Proof. Apply the construction in the proof of Lemma 8.34, but use the stronger induction hypothesis that $\mathbb{L}'_t \subseteq \mathbb{L}_t$ for $\mathbb{L}'_t$ consisting of non-overlapping subconfigurations.

For introduction moves, if the L-configuration $\mathbb{L}'_{t+1}$ is non-overlapping then so is $\mathbb{L}'_t = \mathbb{L}'_{t+1} \setminus v\langle \mathit{pred}(v)\rangle$.

For merger moves $u\langle U\rangle = \mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$, by the induction hypothesis we have $v\langle V\rangle, w\langle W\rangle \notin \mathbb{L}'_{t+1}$, since $\mathbb{L}'_{t+1}$ is non-overlapping and $v\langle V\rangle$ and $w\langle W\rangle$ are covered by $u\langle U\rangle$ by Proposition 8.39. For the same reason $\mathbb{L}'_t$ must be non-overlapping, since we just swap $u\langle U\rangle$ for $v\langle V\rangle$ and $w\langle W\rangle$ with $\mathit{cover}(v\langle V\rangle) \mathbin{\dot\cup} \mathit{cover}(w\langle W\rangle) = \mathit{cover}(u\langle U\rangle)$. (Naturally enough, though, the intermediate L-configurations $\mathbb{L}'_{t+1/3}$ and $\mathbb{L}'_{t+2/3}$, where we merge and erase, will be overlapping.)

Also, any subconfiguration $v\langle V\rangle$ occurs only in one merger, after which it is immediately erased. For at all times $t^* > t$ after which $v\langle V\rangle$ was erased from $\mathcal{L}'$ directly after the first merger move involving $v\langle V\rangle$, there is a $u\langle U\rangle \succ v\langle V\rangle$ in $\mathbb{L}'_{t^*}$. Since all $\mathbb{L}'_{t^*}$ are non-overlapping, the subconfiguration $v\langle V\rangle$ never appears again (this can easily be formalized by a forward induction argument).

Finally, note that in the reversal-free L-pebbling $\mathcal{L}'$, the cover increases at introduction moves, stays the same at mergers, and (by the construction for mergers) also stays the same for erasures. Hence, $\mathit{cover}(\mathbb{L}'_t)$ grows monotonically with t. □

For any L-configuration, we can find an L-configuration with the same cover but consisting only of non-touching subconfigurations. We will refer to this as a *canonical representation.*

Lemma 8.45. *Let V be any non-empty vertex set in T. Then there exists a unique simple L-configuration $\mathbb{L}'$ such that $\mathit{cover}(\mathbb{L}') = V$ and all subconfigurations in $\mathbb{L}'$ are simple and non-touching.*

We introduce the formal definition of canonical representation before proceeding to give a proof of Lemma 8.45.

Definition 8.46 (Canonical representation). For an arbitrary non-empty set of vertices $V \subseteq V(T)$, we define the *canonical representation* $\mathsf{canon}(V)$ of V to be the unique $\mathbb{L}'$ in Lemma 8.45.

For $\mathbb{L}$ an arbitrary L-configuration, we define $\mathsf{canon}(\mathbb{L})$ to be the canonical representation $\mathbb{L}' = \mathsf{canon}(\mathit{cover}(\mathbb{L}))$ of the vertices covered by $\mathbb{L}$.

Once more, we note that this definition is specific for binary trees. Consider, for instance, the set $V = \{u_1, u_2, s_1, s_2, s_3\}$ in Figure 8.3. Both u_1 and u_2 must be black-pebbled in any $\mathbb{L}$ with $\mathit{cover}(\mathbb{L}) = V$, but there is no way two subconfigurations $u_1\langle U_1\rangle$ and $u_2\langle U_2\rangle$ can be non-touching.

Proof of Lemma 8.45. We first show existence and then uniqueness.

We construct $\mathbb{L}'$ with $\mathit{cover}(\mathbb{L}') = V$ as follows: for each $v \in V$ such that $\mathit{succ}(v) \notin V$ or $v = z$, add the subconfiguration $v\langle W\rangle$, where $W \subseteq T_*^v$ is the maximal set such that for all $w \in W$ it holds that $P_*^w \setminus P_*^v \subseteq V$ but $w \notin V$. By construction, $v\langle W\rangle$ is simple, and applying Proposition 8.37 shows that $\mathit{cover}(\mathbb{L}') = V$.

Clearly, every $u \in V$ is covered by exactly one subconfiguration in $\mathbb{L}'$, so all subconfigurations in $\mathbb{L}'$ must be at least non-overlapping. Also, for all $\mathit{Wh}(\mathbb{L}')$ it holds that $w \notin V$ by construction, so the subconfigurations are non-touching.

To get uniqueness, suppose that $\mathbb{L}$ is any simple L-configuration with the property that $\mathit{cover}(\mathbb{L}) = V$. If $v \in V$ but $\mathit{succ}(v) \notin V$, there must be a black pebble on v in $\mathbb{L}$ by Proposition 8.37. Also, if $w \notin V$ but $\mathit{succ}(w) \in V$, w must be white-pebbled. Thus $\mathit{Bl}(\mathbb{L}') \subseteq \mathit{Bl}(\mathbb{L})$ and $\mathit{Wh}(\mathbb{L}') \subseteq \mathit{Wh}(\mathbb{L})$.

The L-configuration $\mathbb{L}$ cannot have pebbles outside $\mathit{cover}(\mathbb{L}') \cup \mathit{Wh}(\mathbb{L}')$, for if so we would have $\mathit{cover}(\mathbb{L}) \supsetneqq V$ (by the convexity property in Definition 8.36 of subconfigurations and since all subconfigurations are simple). And if $\mathbb{L}$ has pebbles inside $\mathit{cover}(\mathbb{L}') \setminus \big(\mathit{Bl}(\mathbb{L}') \cup \mathit{Wh}(\mathbb{L}')\big)$, there must exist touching subconfigurations in $\mathbb{L}$. Hence, if $\mathbb{L}'$ does not contain touching subconfigurations it holds that $\mathbb{L}' = \mathbb{L}$. □

Note, in particular, that if V is a convex set in T in the sense of Definition 8.36, then $\mathsf{canon}(V)$ is a single subconfiguration.

We use the canonical representation to extend Definition 8.41 to L-configurations.

Definition 8.47. Suppose that $\mathbb{L}$, $\mathbb{L}_1$, $\mathbb{L}_2$ are simple L-configurations.

If $\mathit{cover}(\mathbb{L}_1) = \mathit{cover}(\mathbb{L}_2)$, we say that $\mathbb{L}_1$ and $\mathbb{L}_2$ *coincide* and write $\mathbb{L}_1 \sim \mathbb{L}_2$. $\mathbb{L}$ is *non-overlapping* if all distinct $v\langle V\rangle, u\langle U\rangle \in \mathbb{L}$ are pairwise non-overlapping and *non-touching* if all distinct $v\langle V\rangle, u\langle U\rangle \in \mathbb{L}$ are pairwise non-touching. $\mathbb{L}_1$ and $\mathbb{L}_2$ are *mutually non-overlapping* or *mutually non-touching* if all $v\langle V\rangle \in \mathbb{L}_1$ and $u\langle U\rangle \in \mathbb{L}_2$ are pairwise non-overlapping or non-touching, respectively.

Let $\mathbb{L}' = \mathsf{canon}(\mathbb{L})$ be the canonical representation of $\mathbb{L}$. Then the *boundary* of $\mathbb{L}$ is defined to be $\partial\mathbb{L} = \bigcup_{v\langle V\rangle \in \mathbb{L}'} \partial v\langle V\rangle$, the *interior* is defined to be $int(\mathbb{L}) = \bigcup_{v\langle V\rangle \in \mathbb{L}'} int(v\langle V\rangle)$ and the *closure* is $cl(\mathbb{L}) = \bigcup_{v\langle V\rangle \in \mathbb{L}'} cl(v\langle V\rangle)$.

Observe that $\mathbb{L}_1 = \mathbb{L}_2$ implies $\mathbb{L}_1 \sim \mathbb{L}_2$, but not the other way round. For non-touching L-configurations, however, the two notions are identical. Also, $\mathbb{L} \sim \mathsf{canon}(\mathbb{L})$ by definition.

Example 8.48. Returning to Figure 8.7, if we look at the L-configuration $\mathbb{L} = \bigl\{v_1\langle v_2, v_6\rangle,\ v_4\langle v_8, v_9\rangle,\ v_7\langle\emptyset\rangle\bigr\}$ we have $cover(\mathbb{L}) = \{v_1, v_3, v_4, v_7, v_{14}, v_{15}\}$. Since $v_7\langle\emptyset\rangle$ is covered by $v_1\langle v_2, v_6\rangle$ and the subconfigurations $v_1\langle v_2, v_6\rangle$ and $v_4\langle v_8, v_9\rangle$ are non-touching, we get the canonical representation simply by leaving out $v_7\langle\emptyset\rangle$, i.e., $\mathsf{canon}(\mathbb{L}) = \bigl\{v_1\langle v_2, v_6\rangle, v_4\langle v_8, v_9\rangle\bigr\}$. Using this canonical representation of $\mathbb{L}$, we see that

$$\begin{aligned} \partial\mathbb{L} &= \{v_1, v_2, v_4, v_6, v_8, v_9\}, \\ int(\mathbb{L}) &= \{v_3, v_7, v_{14}, v_{15}\}, \\ cl(\mathbb{L}) &= \{v_1, v_2, v_3, v_4, v_6, v_7, v_8, v_9, v_{14}, v_{15}\}. \end{aligned}$$

The L-configuration $\mathbb{L}$ is overlapping because of $v_7\langle\emptyset\rangle$ and $v_1\langle v_2, v_6\rangle$, but, for instance, $\mathbb{L}_1 = \bigl\{v_1\langle v_2, v_6\rangle,\ v_7\langle\emptyset\rangle\bigr\}$ and $\mathbb{L}_2 = \bigl\{v_4\langle v_8, v_9\rangle\bigr\}$ are mutually non-touching.

As a final preliminary before moving on to part 1 in the proof outline in Section 8.5.1, we collect some properties of the L-pebbling cost function of Definition 8.5.

Proposition 8.49. *Suppose that $\mathbb{L}, \mathbb{L}_1, \ldots, \mathbb{L}_m$ are arbitrary simple L-configurations.*

1. *If $\mathbb{L}_1 \subseteq \mathbb{L}_2$ then $\mathsf{cost}(\mathbb{L}_1) \leq \mathsf{cost}(\mathbb{L}_2)$.*
2. *$\mathsf{cost}(\mathbb{L}_1 \cup \mathbb{L}_2) \leq \mathsf{cost}(\mathbb{L}_1) + \mathsf{cost}(\mathbb{L}_2)$.*
3. *If $\mathbb{L}$ is non-touching, $\mathsf{cost}(\mathbb{L}) = \bigl|Bl(\mathbb{L})\bigr| + \bigl|Wh(\mathbb{L})\bigr| = \bigl|\partial\mathbb{L}\bigr|$.*
4. *If $\mathbb{L}_i$ and $\mathbb{L}_j$ are mutually non-touching for $1 \leq i < j \leq m$, it holds that $\mathsf{cost}\bigl(\bigcup_{i=1}^m \mathbb{L}_i\bigr) = \sum_{i=1}^m \mathsf{cost}(\mathbb{L}_i)$.*
5. *If $\mathbb{L}'_i = \mathsf{canon}(\mathbb{L}_i)$ for $i = 1, \ldots, m$, then $\mathsf{cost}\bigl(\bigcup_{i=1}^m \mathbb{L}'_i\bigr) \leq \mathsf{cost}\bigl(\bigcup_{i=1}^m \mathbb{L}_i\bigr)$.*
6. *If $\mathbb{L}' = \mathsf{canon}(\mathbb{L})$, then $\mathsf{cost}(\mathbb{L} \cup \mathbb{L}') = \mathsf{cost}(\mathbb{L})$, and there is an L-pebbling from $\mathbb{L}$ to $\mathbb{L}'$ which does not cost more than $\mathbb{L}$.*

Proof. Parts 1 and 2 are from Proposition 8.21 on page 111 and were proven there.

For part 3, using Definition 8.47 we see that if $\mathbb{L}$ is non-touching, it holds that $Bl(\mathbb{L}) \cap Wh(\mathbb{L}) = \emptyset$. And if the L-configurations $\mathbb{L}_i$ and $\mathbb{L}_j$ are mutually non-touching, we have $\bigl(Bl(\mathbb{L}_i) \cup Wh(\mathbb{L}_i)\bigr) \cap \bigl(Bl(\mathbb{L}_j) \cup Wh(\mathbb{L}_j)\bigr) = \emptyset$, which shows that each pebbled vertex on the left-hand side in part 4 is counted exactly once on the right-hand side.

Part 5 is again immediate since $Bl(\mathbb{L}'_i) \subseteq Bl(\mathbb{L}_i)$ and $Wh(\mathbb{L}'_i) \subseteq Wh(\mathbb{L}_i)$ for $\mathbb{L}'_i = \mathsf{canon}(\mathbb{L}_i)$ by Proposition 8.37 and the proof of Lemma 8.45.

For part 6, $Bl(\mathbb{L} \cup \mathbb{L}') = Bl(\mathbb{L})$ and $Wh(\mathbb{L} \cup \mathbb{L}') = Wh(\mathbb{L})$, which shows that the cost is the same. We also claim that we can do an L-pebbling from $\mathbb{L}$ to $\mathbb{L}' = \mathsf{canon}(\mathbb{L})$ at no extra cost.

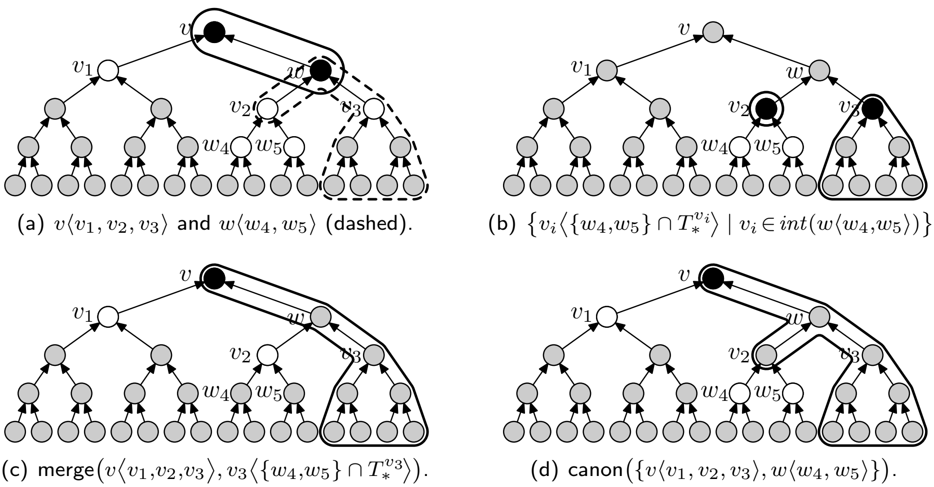
To show this claim, we first note that if $v\langle V\rangle$ and $w\langle W\rangle$ are touching but non-overlapping, we can derive a subconfiguration $u\langle U\rangle$ such that $cover(u\langle U\rangle) = cover(v\langle V\rangle) \cup cover(w\langle W\rangle)$ simply by merging $v\langle V\rangle$ and $w\langle W\rangle$, because either $w \in V$ or $v \in W$. Suppose therefore that $v\langle V\rangle$ and $w\langle W\rangle$ are overlapping and that $w \in T^v$ but $w\langle W\rangle \not\preceq v\langle V\rangle$. Then we can derive a subconfiguration $u\langle U\rangle$ with $cover(u\langle U\rangle) = cover(v\langle V\rangle) \cup cover(w\langle W\rangle)$ and substitute it for $v\langle V\rangle$ and $w\langle W\rangle$ at no extra cost by first deriving $v_i\langle W \cap T_*^{v_i}\rangle$ for all $v_i \in V \cap int(w\langle W\rangle)$ from $w\langle W\rangle$ by reversals, and then merging all $v_i\langle W \cap T_*^{v_i}\rangle$ in turn with $v\langle V\rangle$. The resulting L-configuration $\mathbb{L} \cup \left\{v_i\langle W \cap T_*^{v_i}\rangle \mid v_i \in V \cap int(w\langle W\rangle)\right\} \cup u\langle U\rangle$ costs no more than $\mathbb{L}$, since the only change is that already white-pebbled vertices are also black-pebbled. Finally, erase $v\langle V\rangle$, $w\langle W\rangle$ and all $v_i\langle W \cap T_*^{v_i}\rangle$. Repeating this for all mutually touching subconfigurations, the claim follows by induction. □

A "proof-by-example" pebbling move sequence for part 6 as described above is given in Figure 8.8, with the overlapping subconfigurations $v\langle V\rangle$ and $w\langle W\rangle$ in Figure 8.8(a), the two subconfigurations in $\left\{v_i\langle W \cap T_*^{v_i}\rangle \mid v_i \in V \cap int(w\langle W\rangle)\right\}$ derived by reversals from $w\langle W\rangle$ in Figure 8.8(b), and the two mergers of $v\langle V\rangle$ with these subconfigurations in Figures 8.8(c) and 8.8(d) leading to the canonical representation $\mathsf{canon}\bigl(\{v\langle V\rangle, w\langle W\rangle\}\bigr)$.

8.5.4 Non-Overlapping Labelled Pebblings and Projections

In this subsection we turn to part 1 in the outline of the proof of Lemma 8.31 in Section 8.5.1. From now on we will assume without loss of generality (in view of Lemma 8.40) that all L-pebblings operate with simple subconfigurations only (Definition 8.11), and that they are non-redundant in the sense of Lemma 8.34.

Parts 5 and 6 of Proposition 8.49 tell us that for any given set of vertices, the cheapest way of covering these vertices is to use canonical L-configurations, and that if $\mathbb{L}$ is not canonical, it does not cost anything extra to make $\mathbb{L}$ canonical by applying reversals and mergers followed by erasures. We define *non-overlapping pebblings* as L-pebblings which always keep the L-configurations canonical in this way. In a non-overlapping pebbling, each introduction is immediately followed by a merger when possible, each merger is immediately followed by erasures of the merged subconfigurations, and all reversals from a subconfiguration $u\langle U\rangle$ are performed in sequence after which $u\langle U\rangle$ is erased. We refer to these merger-and-erasures and reversals-and-erasure moves as *expansions* and *implosions*, respectively.



(a) $v\langle v_1, v_2, v_3\rangle$ and $w\langle w_4, w_5\rangle$ (dashed).

(b) $\{v_i\langle\{w_4,w_5\} \cap T_*^{v_i}\rangle \mid v_i \in int(w\langle w_4,w_5\rangle)\}$.

(c) $\mathsf{merge}(v\langle v_1,v_2,v_3\rangle, v_3\langle\{w_4,w_5\} \cap T_*^{v_3}\rangle)$.

(d) $\mathsf{canon}(\{v\langle v_1, v_2, v_3\rangle, w\langle w_4, w_5\rangle\})$.

Figure 8.8: Illustration of canonizing pebbling in Proposition 8.49, part 6.

Definition 8.50 (Non-overlapping pebbling). A *non-overlapping L-pebbling $\mathcal{L}$* is a sequence of the following types of moves.

Introduction $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle pred(v)\rangle$, for $v\langle pred(v)\rangle \not\preceq \mathbb{L}_t$ and $\mathbb{L}_t$ non-touching.

Expansion $\mathbb{L}_{t+3} = \big(\mathbb{L}_t \cup \mathsf{merge}(u\langle U\rangle, v\langle V\rangle)\big) \setminus \{u\langle U\rangle, v\langle V\rangle\}$ for $u\langle U\rangle$, $v\langle V\rangle \in \mathbb{L}_t$ and $\mathbb{L}_t$ non-overlapping.

Implosion $\mathbb{L}_{t+m+1} = \big(\mathbb{L}_t \setminus u\langle U\rangle\big) \cup \mathbb{M}$ for $\mathbb{L}_t$ and $\mathbb{M} = \big\{v_i\langle V_i\rangle \mid i \in [m]\big\}$ non-touching, and $\mathbb{M} \preceq u\langle U\rangle \in \mathbb{L}_t$.

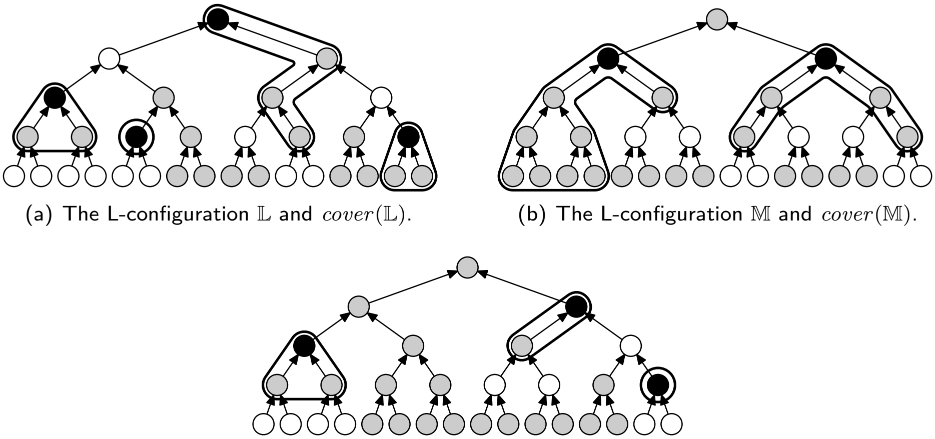
For technical reasons, it will be convenient to allow trivial implosion moves where $\mathbb{M} = u\langle U\rangle$. We say that $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is a *nontrivial implosion* if $\mathbb{M} \prec u\langle U\rangle$. Observe that after introduction and expansion the resulting L-configuration is non-overlapping, and after implosion the L-configuration is non-touching.

We want to prove that without loss of generality we can assume L-pebblings to be non-overlapping. The notation in the proof of this fact is simplified by introducing *projections*.

Definition 8.51 (Projection). Let $u\langle U\rangle, v\langle V\rangle$ be arbitrary subconfigurations, $\mathbb{L}$ an arbitrary L-configuration, and $\mathbb{M}$ an arbitrary non-touching L-configuration.

If $u\langle U\rangle$ and $v\langle V\rangle$ are overlapping, the *projection* of $u\langle U\rangle$ on $v\langle V\rangle$ is defined as $\mathsf{proj}_{v\langle V\rangle}(u\langle U\rangle) = \mathsf{canon}(cover(u\langle U\rangle) \cap cover(v\langle V\rangle))$, i.e., the unique subconfiguration $w\langle W\rangle$ such that $cover(w\langle W\rangle) = cover(u\langle U\rangle) \cap cover(v\langle V\rangle)$. If $u\langle U\rangle$ and $v\langle V\rangle$ are non-overlapping, we define $\mathsf{proj}_{v\langle V\rangle}(u\langle U\rangle) = \emptyset$.

The projection of $u\langle U\rangle$ on $\mathbb{M}$ is $\mathsf{proj}_{\mathbb{M}}(u\langle U\rangle) = \bigcup_{v\langle V\rangle \in \mathbb{M}} \mathsf{proj}_{v\langle V\rangle}(u\langle U\rangle)$, and $\mathsf{proj}_{\mathbb{M}}(\mathbb{L}) = \bigcup_{u\langle U\rangle \in \mathbb{L}} \mathsf{proj}_{\mathbb{M}}(u\langle U\rangle)$.



(a) The L-configuration $\mathbb{L}$ and $cover(\mathbb{L})$.

(b) The L-configuration $\mathbb{M}$ and $cover(\mathbb{M})$.

(c) The projection $\mathsf{proj}_{\mathbb{M}}(\mathbb{L})$ with cover.

Figure 8.9: Example L-configurations $\mathbb{L}$ and $\mathbb{M}$ and projection $\mathsf{proj}_{\mathbb{M}}(\mathbb{L})$.

In order to grasp this definition, it might be helpful to study the example in Figure 8.9. Note, in particular, that if $u\langle U\rangle \preceq v\langle V\rangle$, then $\mathsf{proj}_{v\langle V\rangle}(u\langle U\rangle) = u\langle U\rangle$. Here and in the following, we adopt the convention that projections resulting in the undefined subconfiguration $\emptyset$ are implicitly eliminated from all L-configurations.

We will need a technical lemma relating the pebbles in an L-configuration with those in its projection. Once deciphered, the statements in the lemma are fairly obvious, and the proof is just an exercise in applying the definitions so far in this section. We recommend the reader to look at the projections in (the right subtree of) the tree in Figure 8.9 and verify what the lemma says for this example.

Lemma 8.52. *Let $\mathbb{L}$ be any L-configuration and $\mathbb{M}$ any non-touching L-configuration, and let $\mathbb{L}_p = \mathsf{proj}_{\mathbb{M}}(\mathbb{L})$ be the projection of $\mathbb{L}$ on $\mathbb{M}$. Suppose that v is a vertex that is pebbled in $\mathbb{L}_p$ but not in $\mathbb{L}$, i.e., $v \in \bigl(Bl(\mathbb{L}_p) \cup Wh(\mathbb{L}_p)\bigr) \setminus \bigl(Bl(\mathbb{L}) \cup Wh(\mathbb{L})\bigr)$. Then the following hold:*

1. *The vertex v is on the boundary of $\mathbb{M}$, i.e., $v \in \partial\mathbb{M}$.*
2. *The pebble on the vertex v has the same colour in $\mathbb{L}_p$ and $\mathbb{M}$, i.e., either $v \in Bl(\mathbb{L}_p) \cap Bl(\mathbb{M})$ or $v \in Wh(\mathbb{L}_p) \cap Wh(\mathbb{M})$.*
3. *There is a subconfiguration $w_L\bigl\langle W_L\bigr\rangle \in \mathbb{L}$ such that $v \in int\bigl(w_L\bigl\langle W_L\bigr\rangle\bigr)$.*

Proof. If $v \in Bl(\mathbb{L}_p) \cup Wh(\mathbb{L}_p)$, by Definition 8.51 there are $w_L\langle W_L\rangle \in \mathbb{L}$ and $w_M\langle W_M\rangle \in \mathbb{M}$ with $\mathsf{proj}_{w_M\langle W_M\rangle}(w_L\langle W_L\rangle) = u\langle U\rangle$ such that $v \in \{u\} \cup U$. We remark that since $\mathbb{M}$ is non-touching, $\mathsf{canon}(\mathbb{M}) = \mathbb{M}$ and, in particular, $\partial\mathbb{M} = Bl(\mathbb{M}) \cup Wh(\mathbb{M})$. We make a case analysis depending on the colour of the pebble on v.

1. Suppose $v = u$, i.e., that v is black-pebbled in $\mathbb{L}_p$. Then

$$v \in \mathit{cover}(u\langle U\rangle) = \mathit{cover}(w_L\langle W_L\rangle) \cap \mathit{cover}(w_M\langle W_M\rangle) \tag{8.18}$$

and

$$\mathit{succ}(v) \notin \mathit{cover}(u\langle U\rangle) = \mathit{cover}(w_L\langle W_L\rangle) \cap \mathit{cover}(w_M\langle W_M\rangle) \tag{8.19}$$

by the proof of Lemma 8.45. But $\mathit{succ}(v) \in \mathit{cover}(w_L\langle W_L\rangle)$, since otherwise $v = w_L \in \mathit{Bl}(\mathbb{L})$ by Proposition 8.37, which is contrary to assumption. Thus for (8.19) to hold we must have $\mathit{succ}(v) \notin \mathit{cover}(w_M\langle W_M\rangle)$, so $v = w_M \in \mathit{Bl}(\mathbb{M}) \subseteq \partial\mathbb{M}$. Since v is not pebbled in $\mathbb{L}_p$, in particular we have $v \notin \{w_L\} \cup W_L = \partial w_L\langle W_L\rangle$, and combining this with (8.18) we see that $v \in \mathit{cover}(w_L\langle W_L\rangle) \setminus \partial w_L\langle W_L\rangle = \mathit{int}(w_L\langle W_L\rangle)$.

2. Suppose that $v \in U$, i.e., that v is white-pebbled in $\mathbb{L}_p$. Then

$$v \notin \mathit{cover}(w_L\langle W_L\rangle) \cap \mathit{cover}(w_M\langle W_M\rangle) \tag{8.20}$$

and

$$\mathit{succ}(v) \in \mathit{cover}(w_L\langle W_L\rangle) \cap \mathit{cover}(w_M\langle W_M\rangle) \tag{8.21}$$

by wholly analogous reasoning. We have that $v \in \mathit{cover}(w_L\langle W_L\rangle)$ since otherwise $v \in \mathit{Wh}(\mathbb{L})$ contrary to assumption, so it must hold that $v \notin \mathit{cover}(w_M\langle W_M\rangle)$. Hence, $v \in \mathit{Wh}(\mathbb{M}) \subseteq \partial\mathbb{M}$ and $v \in \mathit{cover}(w_L\langle W_L\rangle) \setminus \partial w_L\langle W_L\rangle = \mathit{int}(w_L\langle W_L\rangle)$. □

This proves the lemma.

The next proposition says that any L-configuration $\mathbb{L}$ can be written as a disjoint union of the sets of subconfigurations of $\mathbb{L}$ covered by each subconfiguration in $\mathsf{canon}(\mathbb{L})$, and that the cost of $\mathbb{L}$ is the sum of the costs of the sub-L-configurations in this disjoint union. This statement, too, is obvious once deciphered, and the proof is immediate from Definition 8.51, (the proof of) Lemma 8.45 and Proposition 8.49, parts 4 and 5.

Proposition 8.53. *Let* $\mathbb{L}' = \mathsf{canon}(\mathbb{L})$. *Then it holds that* $\mathbb{L}$ *is a disjoint union of the sets* $\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}) = \{u\langle U\rangle \mid v\langle V\rangle \succeq u\langle U\rangle \in \mathbb{L}\}$ *for all* $v\langle V\rangle \in \mathbb{L}'$. *Also,* $\mathsf{cost}(\mathbb{L}) = \sum_{v\langle V\rangle\in\mathbb{L}'} \mathsf{cost}(\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}))$, *and for all subconfigurations* $v\langle V\rangle \in \mathbb{L}'$ *it holds that* $\mathsf{cost}(v\langle V\rangle) \leq \mathsf{cost}(\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}))$.

Example 8.54. As we saw in Example 8.48, for $\mathbb{L} = \{v_1\langle v_2, v_6\rangle, v_4\langle v_8, v_9\rangle, v_7\langle\emptyset\rangle\}$ in Figure 8.7 we have the canonical representation $\mathsf{canon}(\mathbb{L}) = \{v_1\langle v_2, v_6\rangle, v_4\langle v_8, v_9\rangle\}$. Trivially, $\mathbb{L}$ can be written as the disjoint union of

$$\mathbb{L}_1 = \mathsf{proj}_{v_1\langle v_2,v_6\rangle}(\mathbb{L}) = \{v_1\langle v_2, v_6\rangle,\ v_7\langle\emptyset\rangle\} \tag{8.22}$$

and

$$\mathbb{L}_2 = \mathsf{proj}_{v_4\langle v_8,v_9\rangle}(\mathbb{L}) = \big\{v_4\langle v_8, v_9\rangle\big\} \tag{8.23}$$

and it holds that $\mathit{cost}(\mathbb{L}) = \mathit{cost}(\mathbb{L}_1) + \mathit{cost}(\mathbb{L}_2)$.

Using Definition 8.51 and Propositions 8.49 and 8.53, we can prove that for every overlapping L-pebbling we can find a non-overlapping pebbling which is at least as good and at least as cheap.

Lemma 8.55. *Suppose that $\mathcal{L}$ is an arbitrary complete L-pebbling of T. Then from $\mathcal{L}$ we can construct a non-overlapping complete L-pebbling $\mathcal{L}'$ of T such that $\mathit{cost}(\mathcal{L}') \leq \mathit{cost}(\mathcal{L})$.*

Proof. Given $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$, we create the "backbone" $\mathcal{L}' = \{\mathbb{L}'_0, \ldots, \mathbb{L}'_\tau\}$ of a non-overlapping pebbling by setting $\mathbb{L}'_t = \mathsf{canon}(\mathbb{L}_t)$. Then we have $\mathbb{L}'_0 = \mathbb{L}_0 = \emptyset$ and $\mathbb{L}'_\tau = \mathsf{canon}(\mathbb{L}_\tau) = \mathsf{canon}(z\langle\emptyset\rangle) = z\langle\emptyset\rangle$.

By Proposition 8.49, part 5, $\mathit{cost}(\mathbb{L}'_t) \leq \mathit{cost}(\mathbb{L}_t)$, so we are done if we can fill in the holes in the transitions $\mathbb{L}'_t \rightsquigarrow \mathbb{L}'_{t+1}$ using the non-overlapping moves of Definition 8.50 without paying more than $\max\big\{\mathit{cost}(\mathbb{L}_t), \mathit{cost}(\mathbb{L}_{t+1})\big\}$. This is basically just an exercise in applying Proposition 8.49. Consider the moves $\mathbb{L}_t \rightsquigarrow \mathbb{L}_{t+1}$ in $\mathcal{L}$.

Introduction $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle \mathit{pred}(v)\rangle$: If $v\langle \mathit{pred}(v)\rangle \preceq \mathbb{L}'_t$, set $\mathbb{L}'_{t+1} = \mathbb{L}'_t$. Otherwise, introduce $v\langle \mathit{pred}(v)\rangle$ and canonize by expanding (at most three times) to get $\mathbb{L}'_{t+1} = \mathsf{canon}(\mathbb{L}_{t+1})$. This can be done at cost at most $\mathit{cost}(\mathbb{L}_{t+1})$, since $\mathit{cost}(\mathbb{L}'_t \cup v\langle \mathit{pred}(v)\rangle) \leq \mathit{cost}(\mathbb{L}_{t+1})$ by part 5 of Proposition 8.49 (note that $\mathsf{canon}(v\langle \mathit{pred}(v)\rangle) = v\langle \mathit{pred}(v)\rangle$), and since the canonization does not increase this cost by part 6 of Proposition 8.49.

Merger $\mathbb{L}_{t+1} = \mathbb{L}_t \cup \mathsf{merge}(u\langle U\rangle, v\langle V\rangle)$ for $u\langle U\rangle, v\langle V\rangle \in \mathbb{L}_t$: For merger moves it holds that $\mathbb{L}_{t+1} \sim \mathbb{L}_t$, so set $\mathbb{L}'_{t+1} = \mathbb{L}'_t = \mathsf{canon}(\mathbb{L}_{t+1})$.

Reversal $\mathbb{L}_{t+1} = \mathbb{L}_t \cup \{v\langle V\rangle\}$ for $v\langle V\rangle \prec u\langle U\rangle \in \mathbb{L}_t$: For reversal moves it holds that $\mathbb{L}_{t+1} \sim \mathbb{L}_t$, so set $\mathbb{L}'_{t+1} = \mathbb{L}'_t = \mathsf{canon}(\mathbb{L}_{t+1})$.

Erasure $\mathbb{L}_{t+1} = \mathbb{L}_t \setminus v\langle V\rangle$ for $v\langle V\rangle \in \mathbb{L}_t$: If $v\langle V\rangle \preceq \mathbb{L}_{t+1}$ we have $\mathbb{L}_{t+1} \sim \mathbb{L}_t$ and can set $\mathbb{L}'_{t+1} = \mathbb{L}'_t$, so assume that $v\langle V\rangle \not\preceq \mathbb{L}_{t+1}$.

Since $\mathbb{L}'_t \sim \mathbb{L}_t$ is non-touching, there is a $u\langle U\rangle \in \mathbb{L}'_t$ such that $v\langle V\rangle \preceq u\langle U\rangle$. It follows from Proposition 8.53 that for $w\langle W\rangle \in \mathbb{L}'_t$, $w\langle W\rangle \neq u\langle U\rangle$, we have $\mathsf{proj}_{w\langle W\rangle}(\mathbb{L}_{t+1}) = \mathsf{proj}_{w\langle W\rangle}(\mathbb{L}_t)$. Thus, letting $\mathbb{L}^u_i = \mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_i)$ for $i = t, t+1$, by Proposition 8.49, part 4, it is sufficient to show locally that we can implode $u\langle U\rangle = \mathsf{canon}(\mathbb{L}^u_t) = \mathsf{canon}(\mathbb{L}^u_{t+1} \cup v\langle V\rangle)$ into $\mathbb{M} = \mathsf{canon}(\mathbb{L}^u_{t+1})$ in cost at most $\max\big\{\mathit{cost}(\mathbb{L}^u_{t+1} \cup v\langle V\rangle), \mathit{cost}(\mathbb{L}^u_{t+1})\big\} = \mathit{cost}(\mathbb{L}^u_{t+1} \cup v\langle V\rangle)$. By part 1 of Proposition 8.49, it is enough to check that the inequality $\mathit{cost}(\mathbb{M} \cup u\langle U\rangle) \leq \mathit{cost}(\mathbb{L}^u_{t+1} \cup v\langle V\rangle)$ holds. But this follows from part 5 of the same proposition by setting $\mathbb{L}_1 = \mathbb{L}^u_{t+1} \cup v\langle V\rangle$ with $\mathbb{L}'_1 = \mathsf{canon}(\mathbb{L}_1) = u\langle U\rangle$ and $\mathbb{L}_2 = \mathbb{L}^u_{t+1}$ with $\mathbb{L}'_2 = \mathsf{canon}(\mathbb{L}_2) = \mathbb{M}$.

Eliminating "idle moves" $\mathbb{L}'_{t+1} = \mathbb{L}'_t$, we see that we get a non-overlapping pebbling in accordance with Definition 8.50. □

Lemma 8.55 tells us that as far as pebbling cost is concerned, without loss of generality we may assume that an L-pebbling $\mathcal{L}$ that reaches the subconfiguration $z\langle\emptyset\rangle$ is non-overlapping. This completes part 1 in the proof of Lemma 8.31 sketched in Section 8.5.1.

In what follows, it will sometimes be convenient to consider the L-pebblings as consisting of the "aggregated" expansion and implosion moves in Definition 8.50, and sometimes more convenient to consider each individual merger or reversal in these moves individually as in Definition 8.5. In view of Lemma 8.55, we know that we can switch freely back and forth between these two perspectives.

8.5.5 Projections Preserve Labelled Pebblings

If $\mathcal{L} = \{\mathbb{L}_0, \dots, \mathbb{L}_\tau\}$ is a non-overlapping pebbling ending in an implosion $u\langle U\rangle \rightsquigarrow \mathbb{M}$, it seems natural to try to replace the moves in $\mathcal{L}$ leading to $u\langle U\rangle$ by a reversal-free pebbling reaching $\mathbb{M} \preceq u\langle U\rangle$. Since $u\langle U\rangle$ and $\mathbb{L}_{\tau-1} \setminus u\langle U\rangle$ are mutually non-touching by definition, this substitution should not affect the cost of the pebbling outside $cl(u\langle U\rangle)$ by Proposition 8.53.

We argue that intuitively, one natural candidate for such a substitution pebbling is what we get if we take all L-configurations in $\mathcal{L}$ and project them on $\mathbb{L}_\tau = (\mathbb{L}_{\tau-(m+1)} \cup \mathbb{M}) \setminus u\langle U\rangle$. To show that this idea makes sense, we establish as a first step that projections preserve merger moves.

Proposition 8.56. *Suppose that $\mathbb{M}$ is a non-touching L-configuration and that $v\langle V\rangle$ and $w\langle W\rangle$ are mergeable with $w \in V$. Then if $\mathsf{proj}_{\mathbb{M}}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) \neq \mathsf{proj}_{\mathbb{M}}(\{v\langle V\rangle, w\langle W\rangle\})$ it holds that $\mathsf{proj}_{\mathbb{M}}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle))$ can be derived from $\mathsf{proj}_{\mathbb{M}}(\{v\langle V\rangle, w\langle W\rangle\})$ by a single merger on w.*

Proof. Consider the merger vertex w. For each $u\langle U\rangle \in \mathbb{M}$ there are four possibilities for w:

1. $w \in \bigcup_{x \in U} T^x$,
2. $w \in P^u$,
3. $w \in T \setminus \bigl(T^u \cup P^u\bigr)$,
4. $w \in int(u\langle U\rangle)$.

See Figure 8.10 for a schematic illustration.

For all $u\langle U\rangle \in \mathbb{M}$ such that $w \notin int(u\langle U\rangle)$, i.e., the first three cases, it is straightforward, if tedious, to verify that $\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$ projects the same subconfigurations on $u\langle U\rangle$ as do $v\langle V\rangle$ and $w\langle W\rangle$ together. In the fourth case, the change in projection corresponds to exactly one merger move, and since $\mathbb{M}$ is non-touching, there is at most one $u\langle U\rangle \in \mathbb{M}$ for which this case applies.

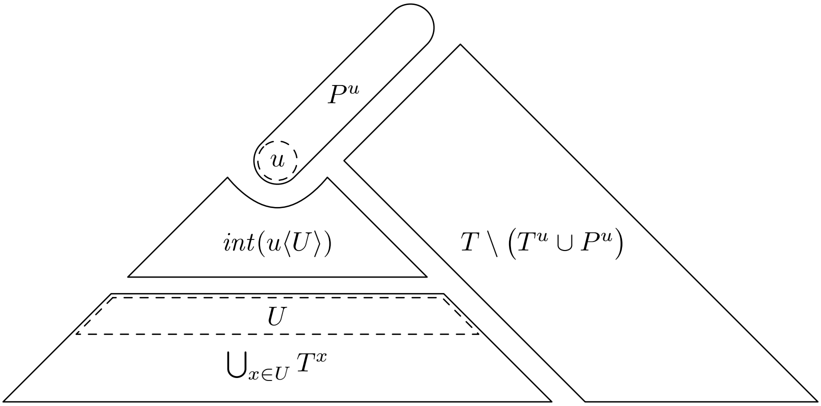


Figure 8.10: Illustration of cases in proof that projections preserve mergers.

We prove these statements by analyzing the cases above one by one, using in the analysis that $\mathit{cover}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) = \mathit{cover}(v\langle V\rangle) \mathbin{\dot\cup} \mathit{cover}(w\langle W\rangle)$ (Proposition 8.39).

1. $w \in \bigcup_{x\in U} T^x$: By Proposition 8.37, $\mathit{cover}(w\langle W\rangle) \subseteq T^w$, and since by assumption it holds that $T^w \subseteq \bigcup_{x\in U} T^x$, it follows that

   $$\mathit{cover}(w\langle W\rangle) \cap \mathit{cover}(u\langle U\rangle) \subseteq T^w \cap \bigl(T^u \setminus \textstyle\bigcup_{x\in U} T^x\bigr) = \emptyset \tag{8.24}$$

   and hence

   $$\begin{aligned}&\mathit{cover}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) \cap \mathit{cover}(u\langle U\rangle) = \\ &\quad= \bigl(\mathit{cover}(v\langle V\rangle) \mathbin{\dot\cup} \mathit{cover}(w\langle W\rangle)\bigr) \cap \mathit{cover}(u\langle U\rangle) = \\ &\qquad\qquad= \mathit{cover}(v\langle V\rangle) \cap \mathit{cover}(u\langle U\rangle) \ .\end{aligned} \tag{8.25}$$

   Consequently, $\mathsf{proj}_{u\langle U\rangle}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) = \mathsf{proj}_{u\langle U\rangle}(v\langle V\rangle)$ according to Definition 8.51, so the merger does not change the projection.

2. $w \in P^u$: Then $\mathit{cover}(u\langle U\rangle) \subseteq T^u \subseteq T^w$, so

   $$\begin{aligned}\mathit{cover}(v\langle V\rangle) \cap \mathit{cover}(u\langle U\rangle) = \bigl(T^v \setminus \textstyle\bigcup_{x\in V} T^x\bigr) \cap \mathit{cover}(u\langle U\rangle) \subseteq \\ \subseteq (T^v \setminus T^w) \cap T^u = \emptyset\end{aligned} \tag{8.26}$$

   and $\mathsf{proj}_{u\langle U\rangle}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) = \mathsf{proj}_{u\langle U\rangle}(w\langle W\rangle)$.

3. $w \in T \setminus \bigl(T^u \cup P^u\bigr)$: Since $\mathit{cover}(w\langle W\rangle) \subseteq T^w$ and $\mathit{cover}(u\langle U\rangle) \subseteq T^u$, in this case we have $\mathit{cover}(w\langle W\rangle) \cap \mathit{cover}(u\langle U\rangle) \subseteq T^w \cap T^u = \emptyset$, and again it holds that $\mathsf{proj}_{u\langle U\rangle}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) = \mathsf{proj}_{u\langle U\rangle}(v\langle V\rangle)$.

4. $w \in \mathit{int}(u\langle U\rangle)$: Note that this implies that $\mathit{cover}(v\langle V\rangle) \cap \mathit{cover}(u\langle U\rangle) \neq \emptyset$ and $\mathit{cover}(w\langle W\rangle) \cap \mathit{cover}(u\langle U\rangle) \neq \emptyset$, which means that the projected subconfigurations $\mathsf{proj}_{u\langle U\rangle}(v\langle V\rangle)$ and $\mathsf{proj}_{u\langle U\rangle}(w\langle W\rangle)$ both exist. Using simple set arithmetic we get that

$$\begin{aligned} &\mathit{cover}\big(\mathsf{proj}_{u\langle U\rangle}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle))\big) = \\ &\quad = \mathit{cover}\big(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)\big) \cap \mathit{cover}(u\langle U\rangle) \\ &\quad = \big(\mathit{cover}(v\langle V\rangle) \mathbin{\dot\cup} \mathit{cover}(w\langle W\rangle)\big) \cap \mathit{cover}(u\langle U\rangle) \\ &\quad = \big(\mathit{cover}(v\langle V\rangle) \cap \mathit{cover}(u\langle U\rangle)\big) \\ &\qquad\qquad \mathbin{\dot\cup} \big(\mathit{cover}(w\langle W\rangle) \cap \mathit{cover}(u\langle U\rangle)\big) \\ &\quad = \mathit{cover}(\mathsf{proj}_{u\langle V\rangle}(v\langle V\rangle)) \mathbin{\dot\cup} \mathit{cover}(\mathsf{proj}_{u\langle V\rangle}(w\langle W\rangle)) \end{aligned} \tag{8.27}$$

and applying Proposition 8.39 we see that indeed

$$\begin{aligned} &\mathsf{proj}_{u\langle U\rangle}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) = \\ &\qquad = \mathsf{merge}(\mathsf{proj}_{u\langle V\rangle}(v\langle V\rangle), \mathsf{proj}_{u\langle V\rangle}(w\langle W\rangle)) \ , \end{aligned} \tag{8.28}$$

i.e., the projection of $\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$ is derivable in one merger step from the projections of $v\langle V\rangle$ and $w\langle W\rangle$ as claimed.

It follows that either $\mathsf{proj}_{\mathbb{M}}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle)) = \mathsf{proj}_{\mathbb{M}}(\{v\langle V\rangle, w\langle W\rangle\})$, if there are no $u\langle U\rangle \in \mathbb{M}$ such that $w \in \mathit{int}(u\langle U\rangle)$, or $\mathsf{proj}_{\mathbb{M}}(\mathsf{merge}(v\langle V\rangle, w\langle W\rangle))$ can be derived from $\mathsf{proj}_{\mathbb{M}}(\{v\langle V\rangle, w\langle W\rangle\})$ by a single merger move for the unique $u\langle U\rangle \in \mathbb{M}$ such that $w \in \mathit{int}(u\langle U\rangle)$. □

The other L-pebbling moves can also be taken care of easily, and we show next that projecting any L-pebbling on any non-touching L-configuration $\mathbb{M}$, we get a legal L-pebbling inside the closure $\mathit{cl}(\mathbb{M})$ (modulo some technical details). In particular, this holds for the non-overlapping pebblings of Definition 8.50. This is part 2 in our proof outline.

Lemma 8.57. *For an arbitrary L-pebbling $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$ and a non-touching L-configuration $\mathbb{M}$, let $\mathsf{proj}_{\mathbb{M}}(\mathcal{L}) = \{\mathbb{L}'_0, \ldots, \mathbb{L}'_\tau\}$ for $\mathbb{L}'_t = \mathsf{proj}_{\mathbb{M}}(\mathbb{L}_t)$. Then $\mathsf{proj}_{\mathbb{M}}(\mathcal{L})$ is a legal L-pebbling if we eliminate idle moves $\mathbb{L}'_{t+1} = \mathbb{L}'_t$ and take care of that one reversal or erasure $\mathbb{L}_t \rightsquigarrow \mathbb{L}_{t+1}$ in $\mathcal{L}$ may correspond to a sequence of reversals or erasures, respectively, in $\mathsf{proj}_{\mathbb{M}}(\mathcal{L})$. Legalizing $\mathsf{proj}_{\mathbb{M}}(\mathcal{L})$ by performing these moves one by one does not affect the pebbling cost, i.e., $\mathsf{cost}(\mathsf{proj}_{\mathbb{M}}(\mathcal{L})) = \max_{t\in\tau}\big\{\mathsf{cost}(\mathsf{proj}_{\mathbb{M}}(\mathbb{L}_t))\big\}$. Also, if $\mathcal{L}$ does not contain any reversals, then neither does $\mathsf{proj}_{\mathbb{M}}(\mathcal{L})$.*

Proof. By induction over the pebbling moves $\mathbb{L}_t \rightsquigarrow \mathbb{L}_{t+1}$ in $\mathcal{L}$. Case analysis:

Introduction If $v\langle \mathit{pred}(v)\rangle \not\preceq \mathbb{M}$ the projection does not change, and otherwise adding $v\langle \mathit{pred}(v)\rangle = \mathsf{proj}_{\mathbb{M}}(v\langle \mathit{pred}(v)\rangle)$ is a legal introduction move.

Merger Suppose $u\langle U\rangle = \mathsf{merge}(v\langle V\rangle, w\langle W\rangle)$. Clearly, $\mathbb{L}_t \setminus \bigl\{u\langle U\rangle, v\langle V\rangle, w\langle W\rangle\bigr\} = \mathbb{L}_{t+1} \setminus \bigl\{u\langle U\rangle, v\langle V\rangle, w\langle W\rangle\bigr\}$, so the only subconfigurations for which the projections can change are $u\langle U\rangle$, $v\langle V\rangle$, and $w\langle W\rangle$. This is Proposition 8.56.

Reversal If $v\langle V\rangle$ is derived from $u\langle U\rangle$ by reversal, we have $v\langle V\rangle \preceq u\langle U\rangle$, or, equivalently, $\mathit{cover}(v\langle V\rangle) \subseteq \mathit{cover}(u\langle U\rangle)$. Then

$$\begin{aligned}\mathit{cover}(\mathsf{proj}_{\mathbb{M}}(v\langle V\rangle)) = \mathit{cover}(v\langle V\rangle) \cap \mathit{cover}(\mathbb{M}) \subseteq\\ \subseteq \mathit{cover}(u\langle U\rangle) \cap \mathit{cover}(\mathbb{M}) = \mathit{cover}(\mathsf{proj}_{\mathbb{M}}(u\langle U\rangle))\ ,\end{aligned} \tag{8.29}$$

so adding $\mathsf{proj}_{\mathbb{M}}(v\langle V\rangle) \preceq \mathsf{proj}_{\mathbb{M}}(u\langle U\rangle)$ is a sequence of legal reversals. As this sequence of reversals is performed, the pebbling cost increases monotonically by part 1 of Proposition 8.49.

Erasure If $\mathbb{L}_{t+1} = \mathbb{L}_t \setminus \{v\langle V\rangle\}$ for $v\langle V\rangle \in \mathbb{L}_t$, removing $\mathsf{proj}_{\mathbb{M}}(v\langle V\rangle)$ from $\mathbb{L}'_t$ is a sequence of legal erasures. As this sequence of erasures is performed, the pebbling cost decreases monotonically by part 1 of Proposition 8.49.

We see that the cost of this pebbling is $\max_{t\in[\tau]}\bigl\{\mathsf{proj}_{\mathbb{M}}(\mathbb{L}_t)\bigr\}$, and if $\mathcal{L}$ is reversal-free, then so is $\mathsf{proj}_{\mathbb{M}}(\mathcal{L})$, since every move in $\mathcal{L}$ is matched by the same kind of moves in $\mathsf{proj}_{\mathbb{M}}(\mathcal{L})$. □

8.5.6 A First (Failed) Attempt to Eliminate Reversal Moves

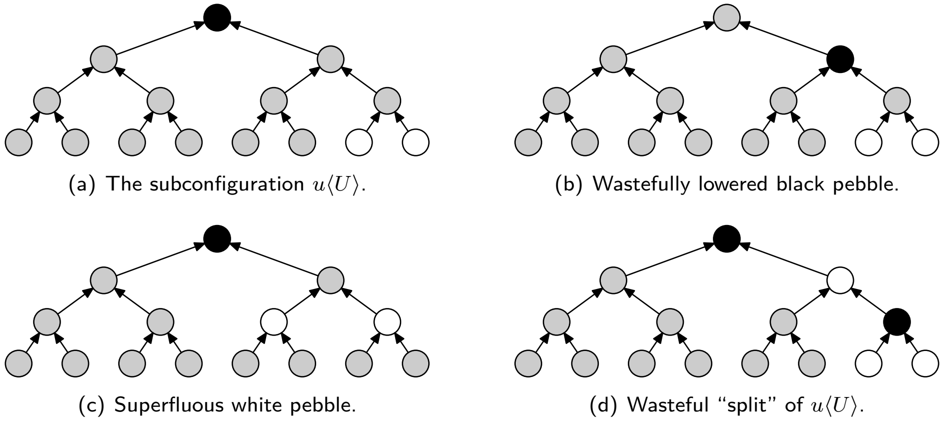
In the light of Lemma 8.57, the following transformation from a non-overlapping pebbling $\mathcal{L}$ to a reversal-free pebbling $\mathcal{L}'$ seems very tempting: by forward induction over the moves in $\mathcal{L}$, replace each implosion $u\langle U\rangle \rightsquigarrow \mathbb{M}$ at time t by a local projection of $\{\mathbb{L}_0, \ldots, \mathbb{L}_t\}$ on $\mathbb{M}$. Since by induction there are no reversals before time t, the projection must be a reversal-free pebbling inside $\mathit{cl}(\mathbb{M})$. Doing this for all implosions, we get a globally reversal-free pebbling $\mathcal{L}'$ ending in $z\langle\emptyset\rangle$. This is the transformation described in part 3 of our roadmap for the proof of Lemma 8.31.

There is only one problem. Although we will indeed get a complete L-pebbling of T, it is not true in general that $\mathit{cost}(\mathsf{proj}_{\mathbb{M}}(\mathcal{L})) \leq \mathit{cost}(\mathcal{L})$. For instance, if $v\langle V\rangle \preceq u\langle\emptyset\rangle$ for $V \neq \emptyset$, then $\mathsf{proj}_{v\langle V\rangle}(u\langle\emptyset\rangle) = v\langle V\rangle$ and hence $\mathit{cost}(\mathsf{proj}_{v\langle V\rangle}(u\langle\emptyset\rangle)) = 1 + |V| > \mathit{cost}(u\langle\emptyset\rangle) = 1$. Looking at this counterexample, however, one might argue that having gotten as far as $u\langle\emptyset\rangle$, reversing to the weaker and more expensive configuration $v\langle V\rangle$ should be non-optimal.

What we want to do next is to define formally which reversals are *wasteful* in this sense, and to prove that for pebblings avoiding such wasteful reversals, projection does not increase the pebbling cost.

8.5.7 Pinpointing the Problem: Wasteful Reversal Moves

Since the definition of wastefulness turns out to be quite technical, we first try to give some more intuition for which kind of reversals we disapprove of.



(a) The subconfiguration $u\langle U\rangle$.

(b) Wastefully lowered black pebble.

(c) Superfluous white pebble.

(d) Wasteful "split" of $u\langle U\rangle$.

Figure 8.11: A subconfiguration $u\langle U\rangle$ and three wasteful implosions of $u\langle U\rangle$.

Example 8.58. Consider the subconfiguration $u\langle U\rangle$ in Figure 8.11(a).

1. If $v \in T_*^u$, the reversal move from $u\langle U\rangle$ to $v\langle T_*^v \cap U\rangle$ seems reasonable only if $T_*^v \cap U \subsetneqq U$, i.e., if we get rid of white pebbles by lowering the black pebble from u to v. The reversal in Figure 8.11(b) does not satisfy this, so it appears we should be better off keeping the original, stronger subconfiguration $u\langle U\rangle$ in Figure 8.11(a) instead.

2. Suppose that V is a simple set below u and above U in the sense that $P^u \cap V \neq \emptyset$ for all $u \in U$. Then we approve of the reversal $u\langle U\rangle \rightsquigarrow u\langle V\rangle$ only if for all $w \in V$ it holds that $T^w \cap U \neq \emptyset$. Otherwise, unnecessary white pebbles have been introduced, as in Figure 8.11(c).

3. If $u\langle U\rangle$ is imploded into a non-touching L-configuration $\{v_1\langle V_1\rangle, v_2\langle V_2\rangle\}$ such that, say, $v_2 \in T_*^{v_1}$, it should not be the case that $v_1\big\langle\big(V_1 \setminus P^{v_2}\big) \cup V_2\big\rangle \preceq u\langle U\rangle$, for if so we could have reversed to this stronger subconfiguration instead of $\{v_1\langle V_1\rangle, v_2\langle V_2\rangle\}$ at no extra cost. The implosion in Figure 8.11(d) violates this condition.

The reversals from $u\langle U\rangle$ in figures 8.11(b), 8.11(c) and 8.11(d) are all examples of wasteful implosions for which our reversal-free pebbling $\mathcal{L}'$ constructed by projection may become more expensive than $\mathcal{L}$. Looking at these examples, it is easy to believe that such moves are non-optimal and that it ought to be possible to eliminate them. The formal definition of wastefulness is as follows.

Definition 8.59 (Wasteful implosion). For a non-touching L-configuration $\mathbb{M} \preceq u\langle U\rangle$, the implosion $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is *non-wasteful* if the following hold:

1. For every $v \in \mathit{Bl}(\mathbb{M}) \setminus \{u\}$ there is a $w \in U \cap T^{\mathit{sibl}(v)}$ such that for the path $Q_v = P^w \setminus P_*^{\mathit{succ}(v)}$ starting at w and ending at $\mathit{succ}(v)$ it holds that $Q_v \cap \bigl(\mathit{Bl}(\mathbb{M}) \cup \mathit{Wh}(\mathbb{M})\bigr) = \emptyset$.
2. For every $v \in \mathit{Wh}(\mathbb{M})$ there is a $w \in U \cap T^v$ (possibly equal to v) such that for the path $Q_v = P^w \setminus P_*^v$ from w to v it holds that $Q_v \cap \mathit{Bl}(\mathbb{M}) = \emptyset$.
3. The paths between the vertices $\bigl(\mathit{Bl}(\mathbb{M}) \cup \mathit{Wh}(\mathbb{M})\bigr) \setminus \{u\}$ and (some subset of) $\mathit{Wh}(u\langle U\rangle) = U$ as described above can all be chosen pairwise disjoint, i.e., such that $Q_v \cap Q_{v'} = \emptyset$ if $v \neq v'$.

If $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is not a non-wasteful implosion, it is said to be *wasteful*.

Definition 8.59 identifies the offending reversal moves for which our projective construction of a reversal-free but cheap pebbling fails (by not being cheap enough).

Loosely put, an implosion move $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is non-wasteful if for every black and white pebble in $\mathbb{M}$ we can identify a distinct pebble in $u\langle U\rangle$ "explaining" why the implosion move could potentially be cost-saving. If there is no such correspondence (as in the implosions from Figure 8.11(a) to Figures 8.11(b), 8.11(c), and 8.11(d), which are all wasteful according to Definition 8.59), the implosion intuitively seems non-optimal, and it should be possible to do better by replacing this wasteful implosion by a stronger, non-wasteful one. And if we can assume that all wasteful implosions are changed into non-wasteful ones, it turns out that our projection idea from Section 8.5.6 does the trick!

Continuing according to part 4 in our proof plan, we show that for pebblings without wasteful moves the projective construction works. This is the next lemma. The thornier task of eliminating wasteful implosions is deferred to Section 8.5.8.

Lemma 8.60. *Suppose that $\mathcal{L} = \bigl\{\mathbb{L}_0 = \emptyset, \ldots, \mathbb{L}_{\tau-2}, \mathbb{L}_{\tau-1} = u\langle U\rangle \rightsquigarrow \mathbb{M}\bigr\}$ is an L-pebbling ending with the non-touching L-configuration $\mathbb{M}$ and containing no reversal moves except for a final non-wasteful implosion $u\langle U\rangle \rightsquigarrow \mathbb{M}$. Then it holds that $\mathsf{cost}(\mathsf{proj}_{\mathbb{M}}(\mathcal{L})) \leq \mathsf{cost}(\mathcal{L})$.*

Proof. By Lemma 8.44, without loss of generality we can assume that $\mathit{cover}(\mathbb{L}_t)$ grows monotonically with t until we reach $\mathbb{L}_{\tau-1} = u\langle U\rangle$. This means that, in particular, there will never be any subconfigurations covering vertices outside $\mathit{cover}(u\langle U\rangle)$ during the pebbling (since such subconfigurations would have to be erased in a redundant way), so it holds that $\mathbb{L}_t \preceq u\langle U\rangle$ for all t.

Let $\mathbb{L}'_t = \mathsf{proj}_{\mathbb{M}}(\mathbb{L}_t)$ for all $t < \tau$. By Lemma 8.57, it suffices to show $\mathsf{cost}(\mathbb{L}'_t) \leq \mathsf{cost}(\mathbb{L}_t)$ to get $\mathsf{cost}(\mathsf{proj}_{\mathbb{M}}(\mathcal{L})) \leq \mathsf{cost}(\mathcal{L})$. This is so since we can go from $\mathbb{L}'_t$ to $\mathbb{L}'_{t+1}$ paying at most $\max\bigl\{\mathsf{cost}(\mathbb{L}'_t), \mathsf{cost}(\mathbb{L}'_{t+1})\bigr\}$, and for $\tau - 1$ we have $\mathsf{proj}_{\mathbb{M}}(\mathbb{L}_{\tau-1}) = \mathsf{proj}_{\mathbb{M}}(u\langle U\rangle) = \mathbb{M}$ since $\mathbb{M} \preceq u\langle U\rangle$.

By definition $\mathsf{cost}(\mathbb{L}_t) = \bigl|\mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)\bigr|$, so to prove $\mathsf{cost}(\mathbb{L}'_t) \leq \mathsf{cost}(\mathbb{L}_t)$ it is enough to find for each vertex $v \in \mathit{Bl}(\mathbb{L}'_t) \cup \mathit{Wh}(\mathbb{L}'_t)$ an associated vertex $v_L \in \mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)$ such that $v_L \neq v_L^*$ if $v \neq v^*$. These associated vertices are exactly what Definition 8.59 will help us find.

If $v^* \in \bigl(\mathit{Bl}(\mathbb{L}'_t) \cup \mathit{Wh}(\mathbb{L}'_t)\bigr) \cap \bigl(\mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)\bigr)$, an obvious choice is $v^*_L = v^*$. Suppose therefore that $v \in \bigl(\mathit{Bl}(\mathbb{L}'_t) \cup \mathit{Wh}(\mathbb{L}'_t)\bigr) \setminus \bigl(\mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)\bigr)$. In this case Lemma 8.52 tells us that $v \in \partial\mathbb{M}$, that v has the same colour in $\mathbb{L}'_t$ and $\mathbb{M}$, i.e., either $v \in \mathit{Bl}(\mathbb{L}'_t) \cap \mathit{Bl}(\mathbb{M})$ or $v \in \mathit{Wh}(\mathbb{L}'_t) \cap \mathit{Wh}(\mathbb{M})$, and that there is a $w_v\bigl\langle W_v\bigr\rangle \in \mathbb{L}_t$ such that $v \in \mathit{int}\bigl(w_v\bigl\langle W_v\bigr\rangle\bigr)$, namely the $w_v\bigl\langle W_v\bigr\rangle$ projecting the pebble on v. We choose $v_L \in \mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)$ for such vertices v by first associating a unique $v_u \in U = \mathit{Wh}(u\langle U\rangle)$ to v as follows.

1. If $v \in \mathit{Bl}(\mathbb{L}'_t) \cap \mathit{Bl}(\mathbb{M})$, pick a vertex $v_u \in U \cap T^{\mathit{sibl}(v)}$ and a path $Q_v = P^{v_u} \setminus P^{\mathit{succ}(v)}_*$ from v_u to $\mathit{succ}(v)$ such that $Q_v \cap \bigl(\mathit{Bl}(\mathbb{M}) \cup \mathit{Wh}(\mathbb{M})\bigr) = \emptyset$ as guaranteed by Definition 8.59. For the subconfiguration $w_v\langle W_v\rangle \in \mathbb{L}_t$ projecting the black pebble on v, we must have $\mathit{succ}(v) \in \mathit{cover}\bigl(w_v\bigl\langle W_v\bigr\rangle\bigr)$ since $v \in \mathit{int}\bigl(w_v\bigl\langle W_v\bigr\rangle\bigr)$, and thus $\mathit{succ}(v) \in Q_v \cap \mathit{cover}\bigl(w_v\bigl\langle W_v\bigr\rangle\bigr) \neq \emptyset$.

2. If $v \in \mathit{Wh}(\mathbb{L}'_t) \cap \mathit{Wh}(\mathbb{M})$, pick $v_u \in U \cap T^v$ and $Q_v = P^{v_u} \setminus P^v_*$ such that $Q_v \cap \mathit{Bl}(\mathbb{M}) = \emptyset$ as provided by Definition 8.59. For $w_v\langle W_v\rangle \in \mathbb{L}_t$ projecting the white pebble on v, we have $v \in \mathit{int}\bigl(w_v\bigl\langle W_v\bigr\rangle\bigr) \subseteq \mathit{cover}\bigl(w_v\bigl\langle W_v\bigr\rangle\bigr)$, so $v \in Q_v \cap \mathit{cover}\bigl(w_v\bigl\langle W_v\bigr\rangle\bigr) \neq \emptyset$.

According to Definition 8.59, all the paths Q_v above can be chosen disjoint.

We claim that $W_v \cap Q_v \neq \emptyset$ for all $v \in \bigl(\mathit{Bl}(\mathbb{L}'_t) \cup \mathit{Wh}(\mathbb{L}'_t)\bigr) \setminus \bigl(\mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)\bigr)$. Given this claim, we can choose as our associated vertex $v_L \in \mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)$ for v the (unique) vertex $v_L \in W_v \cap Q_v$. Since all paths Q_v are disjoint, it follows that all such vertices v_L are distinct. They must also be distinct from the vertices $v^* \in \bigl(\mathit{Bl}(\mathbb{L}'_t) \cup \mathit{Wh}(\mathbb{L}'_t)\bigr) \cap \bigl(\mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)\bigr)$. This is so since the path Q_v does not intersect $\mathbb{M}$ except possibly in v, or in formal notation $\bigl(Q_v \setminus \{v\}\bigr) \cap \mathit{cl}(\mathbb{M}) = \emptyset$, and by construction the associated vertex $v_L \in W_v \cap Q_v$ is always distinct from v (since $v \in \mathit{int}(w_v\langle W_v\rangle)$ but by definition $\mathit{int}(w_v\langle W_v\rangle) \cap W_v = \emptyset$). Hence, for all chosen representatives $v_L \in Q_v$ it holds that $v_L \notin \mathit{cl}(\mathbb{M}) \supseteq \mathit{Bl}(\mathbb{L}'_t) \cup \mathit{Wh}(\mathbb{L}'_t)$. Summing this up, for each $v \in \mathit{Bl}(\mathbb{L}'_t) \cup \mathit{Wh}(\mathbb{L}'_t)$ we have found a distinct associated vertex $v_L \in \mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)$, and it follows that $\mathsf{cost}(\mathbb{L}'_t) \leq \mathsf{cost}(\mathbb{L}_t)$.

It remains to prove the claim that $W_v \cap Q_v \neq \emptyset$ for the path Q_v and subconfiguration $w_v\langle W_v\rangle \in \mathbb{L}_t$ such that $v \in \mathit{int}(w_v\langle W_v\rangle)$ found for each $v \in \bigl(\mathit{Bl}(\mathbb{L}'_t) \cup \mathit{Wh}(\mathbb{L}'_t)\bigr) \setminus \bigl(\mathit{Bl}(\mathbb{L}_t) \cup \mathit{Wh}(\mathbb{L}_t)\bigr)$ above. Fix such a triple $\bigl(v, Q_v, w_v\langle W_v\rangle\bigr)$. By construction, $\mathbb{L}'_t \preceq \mathbb{L}_t \preceq u\langle U\rangle$, so in particular $w_v\langle W_v\rangle \preceq u\langle U\rangle$. Recall that we showed above that $Q_v \cap \mathit{cover}\bigl(w_v\bigl\langle W_v\bigr\rangle\bigr) \neq \emptyset$. Furthermore, we have $Q_v \not\subseteq \mathit{cover}(u\langle U\rangle)$ since the lowest vertex in Q_v is a white pebble of $u\langle U\rangle$. Now if $W_v \cap Q_v = \emptyset$ would hold, this would imply by Proposition 8.37 that all of Q_v lies inside $w_v\langle W_v\rangle$, i.e., $Q_v \subseteq \mathit{cover}(w_v\langle W_v\rangle)$. But this yields the contradiction $w_v\langle W_v\rangle \not\preceq u\langle U\rangle$. Thus $W_v \cap Q_v \neq \emptyset$, which proves the claim. The lemma follows. □

We can use Lemma 8.60 to eliminate non-wasteful implosions one by one without increasing the cost, resulting in a reversal-free L-pebbling.

Lemma 8.61. *Let $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$ be a non-overlapping complete L-pebbling of T without wasteful implosions. Then from $\mathcal{L}$ we can construct a complete L-pebbling $\mathcal{L}'$ of T without reversal moves such that $\mathsf{cost}(\mathcal{L}') \leq \mathsf{cost}(\mathcal{L})$.*

Proof. By induction over the number of implosions. Plainly, if we can go from an L-pebbling $\mathcal{L}$ with n non-wasteful implosions to an L-pebbling $\mathcal{L}'$ with $n-1$ non-wasteful implosions and $\mathsf{cost}(\mathcal{L}') \leq \mathsf{cost}(\mathcal{L})$, the lemma follows by the induction principle.

Consider the subpebbling $\mathcal{L}^*$ of $\mathcal{L}$ consisting of the moves up to and including the first implosion. That is, $\mathcal{L}^* = \{\mathbb{L}_0, \ldots, \mathbb{L}_{\tau^*} \rightsquigarrow (\mathbb{L}_{\tau^*} \setminus u\langle U\rangle) \cup \mathbb{M}\}$ is non-overlapping and reversal-free except for a final non-wasteful implosion $u\langle U\rangle \rightsquigarrow \mathbb{M}$.

By Definition 8.51, the L-configuration $\mathbb{L}_{\tau^*}$ is non-touching, and using Proposition 8.53 each $\mathbb{L}_t$, $t < \tau^*$, can be written as a union of mutually non-touching L-configurations $\mathbb{L}_t = \bigcup_{v\langle V\rangle \in \mathbb{L}_{\tau^*}} \mathbb{L}_t^v$ for $\mathbb{L}_t^v = \mathsf{proj}_{v\langle V\rangle}(\mathbb{L}_t)$ such that $\mathsf{cost}(\mathbb{L}_t) = \sum_{v\langle V\rangle \in \mathbb{L}_{\tau^*}} \mathsf{cost}(\mathbb{L}_t^v)$. Appealing to Lemma 8.57, we see that for all $v\langle V\rangle \in \mathbb{L}_{\tau^*}$, it holds that $\mathcal{L}^v = \{\mathbb{L}_0^v, \ldots, \mathbb{L}_{\tau^*-1}^v, \mathbb{L}_{\tau^*}^v = v\langle V\rangle\}$ are pairwise non-touching pebblings without reversals.

Lemma 8.60 now says that locally, the pebbling $\mathcal{L}^u$ corresponding to the imploded subconfiguration $u\langle U\rangle$ can be replaced by a reversal-free pebbling $\mathsf{proj}_{\mathbb{M}}(\mathcal{L}^u)$ without increasing the local pebbling cost. Then Proposition 8.53 says that we can substitute $\mathsf{proj}_{\mathbb{M}}(\mathcal{L}^u)$ for $\mathcal{L}^u$ in $\mathcal{L}^*$ without increasing the *global* pebbling cost.

Doing this local substitution, instead of $\mathcal{L}^*$ we get a reversal-free pebbling ending in the same L-configuration $(\mathbb{L}_{\tau^*} \setminus u\langle U\rangle) \cup \mathbb{M}$, and it is easy to check using Lemma 8.44 and (the proof of) Lemma 8.45 that this reversal-free pebbling can be made non-overlapping. If we concatenate this pebbling with the rest of the pebbling moves in $\mathcal{L} \setminus \mathcal{L}^*$, we have a non-overlapping complete L-pebbling with one less implosion move. □

This concludes part 4 in the proof outline in Section 8.5.1.

8.5.8 Wasteful Reversal Moves Can Be Replaced

All that remains now is to show that in an arbitrary non-overlapping L-pebbling we can always replace wasteful implosions by non-wasteful ones without increasing the pebbling cost by more than a constant factor. It will take a couple of technical lemmas before we get there, but the intuition from Example 8.58 is clear: if $\mathbb{L}_t \rightsquigarrow \mathbb{L}_{t+m+1}$ is a wasteful implosion, we should be able to match this move with a non-wasteful implosion $\mathbb{L}'_t \rightsquigarrow \mathbb{L}'_{t+m+1}$ instead, where $\mathbb{L}'_i \succeq \mathbb{L}_i$ and $\mathsf{cost}(\mathbb{L}'_i) \leq \mathsf{cost}(\mathbb{L}_i)$ for $i = t, t+m+1$. The only thing that complicates the matter is that we may have to pay extra for the transitional L-configurations during the implosion $\mathbb{L}'_t \rightsquigarrow \mathbb{L}'_{t+m+1}$ because of overlapping subconfigurations.

The cornerstone of our proof is the fact that for every wasteful implosion move $u\langle U\rangle \rightsquigarrow \mathbb{L}$, there is a non-wasteful implosion move to $\mathbb{M} \succ \mathbb{L}$ with $\mathsf{cost}(\mathbb{M}) \leq \mathsf{cost}(\mathbb{L})$.

Lemma 8.62. *If $u\langle U\rangle \rightsquigarrow \mathbb{L}$ is a wasteful implosion, then there is a non-touching $\mathbb{M}$ such that $u\langle U\rangle \succeq \mathbb{M} \succ \mathbb{L}$, $\mathsf{cost}(\mathbb{M}) \leq \min\{\mathsf{cost}(u\langle U\rangle), \mathsf{cost}(\mathbb{L})\}$, and $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is a non-wasteful implosion.*

Proof. If $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is a non-wasteful implosion, it holds that $\mathsf{cost}(\mathbb{M}) = \left|Bl(\mathbb{M})\right| + \left|Wh(\mathbb{M})\right| \leq \mathsf{cost}(u\langle U\rangle) = 1 + |U|$, since by Definition 8.59 every $v \in \big(Bl(\mathbb{M}) \cup Wh(\mathbb{M})\big) \setminus \{u\}$ can be associated with a distinct $w \in U$.

We demonstrate that if $u\langle U\rangle \rightsquigarrow \mathbb{L}$ is a wasteful implosion, we can find an $\mathbb{M}$ such that $u\langle U\rangle \succeq \mathbb{M} \succ \mathbb{L}$ and $\mathsf{cost}(\mathbb{M}) \leq \mathsf{cost}(\mathbb{L})$. If $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is also a wasteful implosion, we repeat this construction to obtain L-configurations $\mathbb{M}'$ with $u\langle U\rangle \succeq \mathbb{M}' \succ \mathbb{M}$ and $\mathsf{cost}(\mathbb{M}') \leq \mathsf{cost}(\mathbb{M})$, $\mathbb{M}''$ with $u\langle U\rangle \succeq \mathbb{M}'' \succ \mathbb{M}'$ $\mathsf{cost}(\mathbb{M}'') \leq \mathsf{cost}(\mathbb{M}')$, et cetera. Sooner or later the process must terminate for some $\mathbb{M}^{(k)} \preceq u\langle U\rangle$ such that $u\langle U\rangle \rightsquigarrow \mathbb{M}^k$ is non-wasteful, since the set of covered vertices $\mathit{cover}\big(\mathbb{M}^{(i)}\big)$ grows in every step. If nothing else, we will end up with $\mathbb{M}^{(k)} = u\langle U\rangle$, and by definition the trivial implosion $u\langle U\rangle \rightsquigarrow u\langle U\rangle$ is non-wasteful.

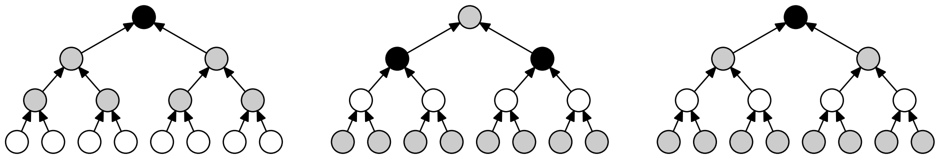
According to Definition 8.59, the configuration $\mathbb{L}$ can be wasteful with respect to $u\langle U\rangle$ in three ways. For the purpose of the case analysis, it appears more natural in this lemma (but only in this lemma) to traverse the paths in T in the reverse direction, so that we move downwards from above.

1. Some black pebble $v \in Bl(\mathbb{L})\setminus\{u\}$ lacks a path. That is, all paths from $\mathit{succ}(v)$ downwards in the sibling subtree $T^{\mathit{sibl}(v)}$ to white pebbles in U intersect with other pebbled vertices in $\mathbb{L}$.

   If $\mathit{succ}(v) \in Wh(\mathbb{L})$ we must have $\mathit{succ}(v) \in \mathit{cover}(u\langle U\rangle)$ by convexity (Definition 8.36 and Proposition 8.37), so we can enlarge the cover by adding the subconfiguration $\mathsf{canon}(\{\mathit{succ}(v)\}) = \mathit{succ}(v)\langle v, \mathit{sibl}(v)\rangle$ to $\mathbb{L}$ and canonize to get $\mathbb{M} = \mathsf{canon}(\mathbb{L} \cup \mathit{succ}(v)\langle v, \mathit{sibl}(v)\rangle) \succ \mathbb{L}$ with $\mathsf{cost}(\mathbb{M}) \leq \mathsf{cost}(\mathbb{L}) + |\{\mathit{sibl}(v)\}| - |\{v, \mathit{succ}(v)\}| < \mathsf{cost}(\mathbb{L})$. We note that this is so since a black and a white pebble on the same vertex "cancel" and can be eliminated by a merger on this vertex. Figure 8.11(d) is an illustration of this case.

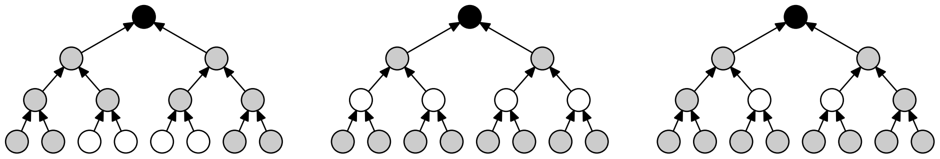
   Otherwise, since $\mathbb{L}$ is non-touching all paths from $\mathit{succ}(v)$ downwards in $T^{\mathit{sibl}(v)}$ are either blocked by $r_1, \ldots, r_m \in Bl(\mathbb{L}) \cap T^{\mathit{sibl}(v)}$ or reach sources in $T^{\mathit{sibl}(v)}$ without passing pebbled vertices (if there are no black pebbles in $T^{\mathit{sibl}(v)}$, we let $m = 0$). By the convexity of $\mathit{cover}(u\langle U\rangle)$, we conclude that $V = T^{\mathit{succ}(v)} \setminus \big(T^v \cup \bigcup_{i\in[m]} T^{r_i}\big) \subseteq \mathit{cover}(u\langle U\rangle)$, so we can add $\mathsf{canon}(V) = \mathit{succ}(v)\langle v, r_1, \ldots, r_m\rangle \preceq u\langle U\rangle$ to $\mathbb{L}$. This move increases the cost only by 1 for the unpebbled vertex $\mathit{succ}(v)$, since the vertices $v, r_1, \ldots, r_m$ are all pebbled. Setting $\mathbb{M} = \mathsf{canon}(\mathbb{L} \cup \mathit{succ}(v)\langle v, r_1, \ldots, r_m\rangle) \succ \mathbb{L}$ removes the pebbles from the black- and white-pebbled vertices $v, r_1, \ldots, r_m$ and decreases the cost by at least 1, so $\mathsf{cost}(\mathbb{M}) \leq \mathsf{cost}(\mathbb{L})$. See Figure 8.12 for a simple example.

2. There is a white pebble $w \in Wh(\mathbb{L})$ such that all paths downwards in T^w either are blocked by $r_1, \ldots, r_m \in Bl(\mathbb{L}) \cap T^w_*$ or reach sources in T^w without passing pebbled vertices. If so, we have $V = T^w \setminus \bigcup_{i\in[m]} T^{r_i} \subseteq \mathit{cover}(u\langle U\rangle)$,



(a) The subconfiguration. (b) Wasteful implosion $\mathbb{L}$. (c) Non-wasteful $\mathbb{M} \succ \mathbb{L}$.

Figure 8.12: Illustration of case 1 in proof of Lemma 8.62.



(a) The subconfiguration. (b) Wasteful implosion $\mathbb{L}$. (c) Non-wasteful $\mathbb{M} \succ \mathbb{L}$.

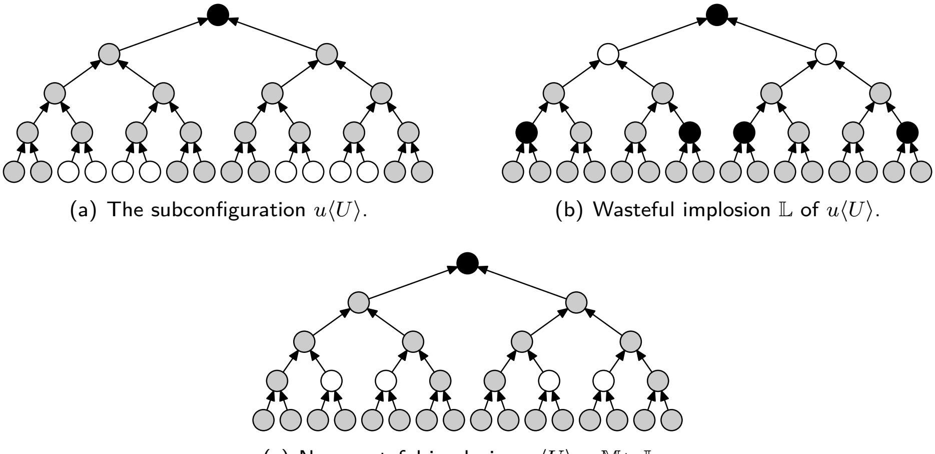
Figure 8.13: Illustration of case 2 in proof of Lemma 8.62.

and we can add $\mathsf{canon}(V) = w\langle r_1, \ldots, r_m\rangle \preceq u\langle U\rangle$ to $\mathbb{L}$ at no extra cost and set $\mathbb{M} = \mathsf{canon}(\mathbb{L} \cup w\langle r_1, \ldots, r_m\rangle) \succ \mathbb{L}$. Here we get a strict inequality $\mathit{cost}(\mathbb{M}) < \mathit{cost}(\mathbb{L})$ since the canonization eliminates at least the pebble on w. This case is illustrated in Figure 8.13.

3. There are paths for all $v \in \bigl(\mathit{Bl}(\mathbb{L}) \cup \mathit{Wh}(\mathbb{L})\bigr) \setminus \{u\}$ to vertices in U in the sense of Definition 8.59, but they cannot be chosen disjoint. Start picking disjoint paths bottom-up from the leaves towards the root so that when we choose a path for a white pebble $v \in \mathit{Wh}(\mathbb{L})$ we have already determined paths for all $w \in \bigl(\mathit{Bl}(\mathbb{L}) \cup \mathit{Wh}(\mathbb{L})\bigr) \cap T_*^v$, and when we choose a path for a black pebble $v \in \mathit{Bl}(\mathbb{L})$ we have already determined paths for all $w \in \bigl(\mathit{Bl}(\mathbb{L}) \cup \mathit{Wh}(\mathbb{L})\bigr) \cap T^{\mathit{sibl}(v)}$, or in fact for all of $T^{\mathit{succ}(v)} \setminus \{v\}$. This can be done since for black pebbles, the vertex $\mathit{sibl}(v)$ itself cannot be black-pebbled in $\mathbb{L}$, for if so there would be no path for v and we would be in case 1. For the same reason, $\mathit{succ}(v)$ is not white-pebbled in $\mathbb{L}$, and then $\mathit{sibl}(v)$ cannot be white-pebbled, nor can $\mathit{succ}(v)$ be black-pebbled, since $\mathbb{L}$ is non-touching.

   At some point we reach a v such that no matter how we choose the paths below, we cannot choose a disjoint path for v. Consider the colour of v.

   (a) v is black. Then there are white pebbles in $U \cap T_*^{\mathit{sibl}(v)}$ reachable from v, but they are all blocked by paths already chosen from black-pebbled vertices $r_1, \ldots, r_m \in \mathit{Bl}(\mathbb{L}) \cap T_*^{\mathit{sibl}(v)}$. (Note that all white pebbles in $T_*^{\mathit{sibl}(v)}$ are located below black pebbles since $\mathbb{L}$ is non-touching, so no paths from white-pebbled vertices in $T_*^{\mathit{sibl}(v)}$ are among the "blocking paths"



(a) The subconfiguration $u\langle U\rangle$.

(b) Wasteful implosion $\mathbb{L}$ of $u\langle U\rangle$.

(c) Non-wasteful implosion $u\langle U\rangle \rightsquigarrow \mathbb{M} \succ \mathbb{L}$.

Figure 8.14: Illustration of case 3 in proof of Lemma 8.62.

for our vertex v.) This means that $\{succ(r_i) \mid i \in [m]\} \subseteq cover(u\langle U\rangle)$ by the convexity of $cover(u\langle U\rangle)$, so we can add all of the subconfigurations $\mathsf{canon}(\{succ(r_i) \mid i \in [m]\}) = \bigl\{succ(r_i)\langle r_i, sibl(r_i)\rangle \mid i \in [m]\bigr\}$ to $\mathbb{L}$ at an additional cost $2m$. By similar reasoning we can also add $succ(v)\langle v, succ(r_1), \ldots, succ(r_m)\rangle$ at a further cost of 1 for the unpebbled vertex $succ(v)$. When we canonize this L-configuration, the black and white pebbles on the vertices $v, r_1, \ldots, r_m, succ(r_1), \ldots, succ(r_m)$ all cancel and disappear and the cost decreases by $2m + 1$, resulting in $\mathbb{M} \succ \mathbb{L}$ with $\mathit{cost}(\mathbb{M}) \leq \mathit{cost}(\mathbb{L})$.

(b) v is white. The construction is analogous. Let the blocking black pebbles in T^v_* be $r_1, \ldots, r_m \in Bl(\mathbb{L}) \cap T^v_*$. Again $succ(r_i)\langle r_i, sibl(r_i)\rangle$, $i \in [m]$, can be added at an extra cost $2m$. Since $succ(r_i)$, $i \in [m]$, block all paths from v we have $T^v \setminus \bigcup_{i\in[m]} T^{succ(r_i)} \subseteq cover(u\langle U\rangle)$, so $v\langle succ(r_1), \ldots, succ(r_m)\rangle$ can be added as well at no additional cost. Canonizing decreases the cost by $2m+1$, which yields an L-configuration $\mathbb{M} \succ \mathbb{L}$ with $\mathit{cost}(\mathbb{M}) < \mathit{cost}(\mathbb{L})$. The transition from Figure 8.14(b) to Figure 8.14(c) is accomplished by applying this procedure twice.

In all cases we can find a non-touching L-configuration $\mathbb{M}$ such that $u\langle U\rangle \succeq \mathbb{M} \succ \mathbb{L}$ and $\mathit{cost}(\mathbb{M}) \leq \mathit{cost}(\mathbb{L})$. The lemma follows by induction. □

The following transitivity property of non-wasteful implosions is an immediate consequence of Definition 8.59.

Observation 8.63. *If $u\langle U\rangle \rightsquigarrow \{v_i\langle V_i\rangle \mid i \in [m]\}$ and $v_i\langle V_i\rangle \rightsquigarrow \mathbb{M}_i$ for $i \in [m]$ are all non-wasteful implosions, then $u\langle U\rangle \rightsquigarrow \{\mathbb{M}_i \mid i \in [m]\}$ is a non-wasteful implosion.*

Proof. For each $i \in [m]$, concatenate the paths from $\mathbb{M}_i$ to $v_i\langle V_i\rangle$ provided by Definition 8.59 with those from $v_i\langle V_i\rangle$ to $u\langle U\rangle$ provided by the same definition. The result is a set of disjoint paths from $\bigcup_{i\in[m]} \mathbb{M}_i$ to $u\langle U\rangle$ as required by Definition 8.59. □

It follows from Observation 8.63 that if $u\langle U\rangle \rightsquigarrow \mathbb{L}$ is a wasteful implosion and $u\langle U\rangle \rightsquigarrow \mathbb{M} \succ \mathbb{L}$ is a corresponding non-wasteful implosion for $\mathbb{M}$ minimal, then all nontrivial "local implosions" from subconfigurations in $\mathbb{M}$ to sets of subconfigurations in $\mathbb{L}$ are wasteful. We formalize this as a lemma.

Lemma 8.64. *Suppose that $u\langle U\rangle \rightsquigarrow \mathbb{L}$ is a wasteful implosion and let $\mathbb{M} \succ \mathbb{L}$ be minimal such that $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is non-wasteful. Then for each $v\langle V\rangle \in \mathbb{M}$ and each non-touching $\mathbb{L}'$ such that $\mathbb{M} \succ \mathbb{L}' \succeq \mathbb{L}$, either $\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}') = v\langle V\rangle$ or $v\langle V\rangle \rightsquigarrow \mathsf{proj}_{v\langle V\rangle}(\mathbb{L}')$ is a wasteful implosion. In particular, for each $v\langle V\rangle \in \mathbb{M}$ it holds that $\mathit{cost}(v\langle V\rangle) \leq \mathit{cost}(\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}'))$.*

Proof. Given a subconfiguration $u\langle U\rangle$ and an L-configuration $\mathbb{L}$ as in the statement of the lemma, we know from Lemma 8.62 that we can find some $\mathbb{M}$ such that $u\langle U\rangle \rightsquigarrow \mathbb{M}$ is a non-wasteful implosion and $u\langle U\rangle \succeq \mathbb{M} \succeq \mathbb{L}$. Pick such an $\mathbb{M}$ which is minimal with respect to $\preceq$. Note that by the definition of implosion moves, $\mathbb{L}$ and $\mathbb{M}$ are non-touching.

Suppose that there is a subconfiguration $v\langle V\rangle \in \mathbb{M}$ and an L-configuration $\mathbb{L}'$ with $\mathbb{M} \succ \mathbb{L}' \succeq \mathbb{L}$ such that $\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}') \prec v\langle V\rangle$ and $v\langle V\rangle \rightsquigarrow \mathsf{proj}_{v\langle V\rangle}(\mathbb{L}')$ is a non-wasteful implosion. Then by the transitivity in Observation 8.63 it holds that $\mathbb{M}' = \big(\mathbb{M} \cup \mathsf{proj}_{v\langle V\rangle}(\mathbb{L}')\big) \setminus v\langle V\rangle \prec \mathbb{M}$ is a non-wasteful implosion of $u\langle U\rangle$. This contradicts the minimality of $\mathbb{M}$.

If $v\langle V\rangle \rightsquigarrow \mathsf{proj}_{v\langle V\rangle}(\mathbb{L}')$ is a wasteful implosion, Lemma 8.62 says that there exists a non-wasteful implosion locally to an L-configuration $\mathbb{M}_v$ with $v\langle V\rangle \succeq \mathbb{M}_v \succ \mathsf{proj}_{v\langle V\rangle}(\mathbb{L}')$ such that $\mathit{cost}(\mathbb{M}_v) \leq \min\big\{\mathit{cost}(v\langle V\rangle), \mathit{cost}(\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}'))\big\}$, and, in particular, $\mathit{cost}(\mathbb{M}_v) \leq \mathit{cost}(\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}'))$. But we have just proven that this non-wasteful $\mathbb{M}_v$ must be identical with $v\langle V\rangle$, so $\mathit{cost}(v\langle V\rangle) \leq \mathit{cost}(\mathsf{proj}_{v\langle V\rangle}(\mathbb{L}'))$. □

Very roughly, the next lemma says that wasteful implosions are preserved under mergers.

Lemma 8.65. *Suppose for $i = 1, 2$ that $u_i\langle U_i\rangle \succeq \mathbb{L}_i$ and $\mathit{cost}(u_i\langle U_i\rangle) \leq \mathit{cost}(\mathbb{L}_i)$ for $\mathbb{L}_i$ non-overlapping, and that $u_1\langle U_1\rangle$ and $u_2\langle U_2\rangle$ are mutually non-overlapping with $u_2 \in U_1$. Then it holds that $\mathit{cost}(\mathsf{merge}(u_1\langle U_1\rangle, u_2\langle U_2\rangle)) \leq \mathit{cost}(\mathbb{L}_1 \cup \mathbb{L}_2)$.*

Proof. The L-configurations $\mathbb{L}_1$ and $\mathbb{L}_2$ must be mutually non-overlapping since they are covered by $u_1\langle U_1\rangle$ and $u_2\langle U_2\rangle$, respectively. The only way that $\mathit{cost}(\mathbb{L}_1 \cup \mathbb{L}_2)$ could be less than $\mathit{cost}(\mathsf{merge}(u_1\langle U_1\rangle, u_2\langle U_2\rangle)) = \mathit{cost}(u_1\langle U_1\rangle) + \mathit{cost}(u_2\langle U_2\rangle) - 1 \leq \mathit{cost}(\mathbb{L}_1) + \mathit{cost}(\mathbb{L}_2) - 1$ is if there were at least two vertices in $\bigcap_{i=1,2}\big(Bl(\mathbb{L}_i) \cup$

$\mathit{Wh}(\mathbb{L}_i)\big)$. But $\mathit{Bl}(\mathbb{L}_i) \cup \mathit{Wh}(\mathbb{L}_i) \subseteq \mathit{cl}(\mathbb{L}_i) \subseteq \mathit{cl}(u_i\langle U_i\rangle)$ since $\mathbb{L}_i \preceq u_i\langle U_i\rangle$ by the assumptions of the lemma, and also by assumption $\mathit{cl}(u_1\langle U_1\rangle) \cap \mathit{cl}(u_1\langle U_1\rangle) = \{u_2\}$ since $u_1\langle U_1\rangle$ and $u_2\langle U_2\rangle$ are mergeable (recall Example 8.43), so this is impossible. □

Combining Lemmas 8.64 and 8.65, we can provide the fifth and final component in the proof of Lemma 8.31, namely, that any non-overlapping L-pebbling $\mathcal{L}$ can be transformed into a pebbling $\mathcal{L}'$ without wasteful implosions such that $\mathcal{L}'$ has asymptotically the same cost as $\mathcal{L}$.

Lemma 8.66. *Suppose that $\mathcal{L}$ is a non-overlapping complete L-pebbling of T. Then we can find a non-overlapping complete L-pebbling $\mathcal{L}'$ of T without wasteful implosions such that $\mathsf{cost}(\mathcal{L}') \leq 2 \cdot \mathsf{cost}(\mathcal{L})$.*

Proof. In this proof, let us assume for simplicity (and without loss of generality what concerns pebbling cost, by the proof of Lemma 8.55) that each introduction, expansion or implosion move in Definition 8.50 takes exactly one time step.

Given a non-overlapping L-pebbling $\mathcal{L}$, we build a non-overlapping L-pebbling $\mathcal{L}'$ without wasteful implosions such that if we let $\mathbb{L}_i \in \mathcal{L}$ denote the starting configuration of the ith move in $\mathcal{L}$, there is a corresponding $\mathbb{L}'_i \in \mathcal{L}'$ such that the following invariants hold:

1. $\mathbb{L}'_i$ is non-touching.
2. $\mathbb{L}'_i \succeq \mathbb{L}_i$.
3. For all $u\langle U\rangle \in \mathbb{L}'_i$, it holds that $\mathsf{cost}(u\langle U\rangle) \leq \mathsf{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_i))$.
4. The cost of the L-pebbling transition from $\mathbb{L}'_{i-1}$ to $\mathbb{L}'_i$ in $\mathcal{L}'$ does not exceed $2 \cdot \max\big\{\mathsf{cost}(\mathbb{L}_{i-1}), \mathsf{cost}(\mathbb{L}_i)\big\}$.

To see that the lemma follows from this, note that invariants 1 and 2 imply that for every $v\langle V\rangle \in \mathbb{L}_i$ there is a $u\langle U\rangle \in \mathbb{L}'_i$ such that $v\langle V\rangle \preceq u\langle U\rangle$. In particular, for $\mathbb{L}_\tau = z\langle\emptyset\rangle$ we have $z\langle\emptyset\rangle \in \mathbb{L}'_\tau$, since $z\langle\emptyset\rangle$ is the maximal element with respect to $\preceq$. Then plugging invariant 3 into Proposition 8.49, part 4, we get

$$\mathsf{cost}(\mathbb{L}'_i) = \sum_{u\langle U\rangle \in \mathbb{L}'_i} \mathsf{cost}(u\langle U\rangle) \leq \sum_{u\langle U\rangle \in \mathbb{L}'_i} \mathsf{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_i)) = \sum_{u\langle U\rangle \in \mathbb{L}'_i} \mathsf{cost}(\{v\langle V\rangle \in \mathbb{L}_i \mid v\langle V\rangle \preceq u\langle U\rangle\}) = \mathsf{cost}(\mathbb{L}_i) \ . \tag{8.30}$$

Using invariant 4 to bound the cost of the pebbling transitions $\mathbb{L}'_{i-1} \rightsquigarrow \mathbb{L}'_i$, we get the desired result $\mathsf{cost}(\mathcal{L}') \leq 2 \cdot \mathsf{cost}(\mathcal{L})$.

The construction is by forward induction over the moves in $\mathcal{L}$. Assume that the invariants hold for $\mathbb{L}_t$ and $\mathbb{L}'_t$.

Introduction $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle pred(v)\rangle$: If $v\langle pred(v)\rangle \preceq \mathbb{L}'_t$ we set $\mathbb{L}'_{t+1} = \mathbb{L}'_t$. For the pebble subconfiguration $u\langle U\rangle \in \mathbb{L}'_t$ such that $v\langle pred(v)\rangle \preceq u\langle U\rangle$, we have $\mathit{cost}(u\langle U\rangle) \leq \mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_t)) \leq \mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_t \cup v\langle pred(v)\rangle))$, and for $u'\left\langle U'\right\rangle \in \mathbb{L}'_t$ distinct from $u\langle U\rangle$ nothing changes. All invariants stay true.

If $v\langle pred(v)\rangle \not\preceq \mathbb{L}'_t$, we introduce $v\langle pred(v)\rangle$ in $\mathcal{L}'$ and expand (at most three times) to get $\mathbb{L}'_{t+1} = \mathsf{canon}(\mathbb{L}'_t \cup v\langle pred(v)\rangle)$. Invariants 1 and 2 obviously hold. We claim that invariant 3 holds with respect to $\mathbb{L}_{t+1}$ instead of $\mathbb{L}_t$ for all subconfigurations in the intermediate L-configurations $\mathbb{L}'$ in the transition $\mathbb{L}'_t \rightsquigarrow \mathbb{L}'_{t+1}$ up to and including $\mathbb{L}'_{t+1} = \mathsf{canon}(\mathbb{L}'_t \cup v\langle pred(v)\rangle)$. This claim yields invariants 3 and 4 for $\mathbb{L}'_{t+1}$.

To prove the claim, observe that invariant 3 holds for $\mathbb{L}'_t \cup v\langle pred(v)\rangle$ with respect to $\mathbb{L}_{t+1} = \mathbb{L}_t \cup v\langle pred(v)\rangle$ by the induction hypothesis and the fact that $\mathsf{proj}_{v\langle pred(v)\rangle}(\mathbb{L}_t \cup v\langle pred(v)\rangle) = v\langle pred(v)\rangle$. Since $\mathbb{L}'_{t+1}$ is obtained by repeated merging of non-overlapping subconfigurations from $\mathbb{L}'_t \cup v\langle pred(v)\rangle$, and since by induction over each such merger these subconfigurations meet the conditions in Lemma 8.65, the claim follows.

Expansion $\mathbb{L}_{t+1} = \big(\mathbb{L}_t \cup \mathsf{merge}(v_1\langle V_1\rangle, v_2\langle V_2\rangle)\big)\setminus\{v_1\langle V_1\rangle, v_2\langle V_2\rangle\}$: By induction it holds that $\mathbb{L}'_t \succeq \mathbb{L}_t \sim \mathbb{L}_{t+1}$, so there is a $u\langle U\rangle \in \mathbb{L}'_t$ such that $v_i\langle V_i\rangle \preceq u\langle U\rangle$ for $i = 1, 2$. For $u'\left\langle U'\right\rangle \in \mathbb{L}'_t$ distinct from $u\langle U\rangle$ there are no changes in the invariants, and if $\mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{t+1})) \geq \mathit{cost}(u\langle U\rangle)$, nothing needs to be done for $u\langle U\rangle$ either and we can set $\mathbb{L}'_{t+1} = \mathbb{L}'_t$.

It can be the case, however, that the expansion within $\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{t+1})$ decreased the cost so that $u\langle U\rangle$ is now too expensive and invariant 3 no longer holds. If so, we implode $u\langle U\rangle$ to a minimal non-wasteful L-configuration $\mathbb{M}_u \succeq \mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{t+1})$ and set $\mathbb{L}'_{t+1} = \big(\mathbb{L}'_t \setminus u\langle U\rangle\big) \cup \mathbb{M}_u$.

Invariants 1 and 2 are immediate. Invariant 3 follows from Lemma 8.64 since $\mathbb{M}_u$ is chosen minimal. Thus, $\mathit{cost}(\mathbb{M}_u) \leq \mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{t+1}))$, and by the induction hypothesis we know that $\mathit{cost}(u\langle U\rangle) \leq \mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_t))$. Using part 1 of Proposition 8.49, we see that the maximal cost in the implosion sequence $\mathbb{L}'_t \rightsquigarrow \mathbb{L}'_{t+1}$ locally inside the closure $cl(u\langle U\rangle)$ is reached in the L-configuration $u\langle U\rangle \cup \mathbb{M}_u$, and using part 2 of Proposition 8.49, this extra cost in the transition from $\mathbb{L}'_t$ to $\mathbb{L}'_{t+1}$ in $\mathcal{L}'$ is at most

$$\begin{aligned}
\mathit{cost}(u\langle U\rangle \cup \mathbb{M}_u) &\leq \mathit{cost}(u\langle U\rangle) + \mathit{cost}(\mathbb{M}_u) \\
&\leq \mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_t)) + \mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{t+1})) \\
&\leq 2 \cdot \max_{i\in\{t,t+1\}} \big\{\mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_i))\big\} \ .
\end{aligned} \tag{8.31}$$

The cost outside $cl(u\langle U\rangle)$ does not change since nothing happens there, so invariant 4 follows.

Implosion $\mathbb{L}_{t+1} = \big(\mathbb{L}_t \setminus v\langle V\rangle\big) \cup \mathbb{M}$ for $\mathbb{M} = \big\{v_i\langle V_i\rangle \mid i \in [m]\big\}$: This case is analogous to the expansion case. By invariants 1 and 2, we know that $v\langle V\rangle$

is covered by some $u\langle U\rangle \in \mathbb{L}'_t$. Nothing happens for $u'\langle U'\rangle \in \mathbb{L}'_t$ distinct from $u\langle U\rangle$, so we can again concentrate on what is going on inside $cl(u\langle U\rangle)$.

If $u\langle U\rangle$ is too expensive with respect to $\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{t+1})$ so that invariant 3 fails, we make a non-wasteful implosion of $u\langle U\rangle$ to an L-configuration $\mathbb{M}_u$ with $u\langle U\rangle \succeq \mathbb{M}_u \succeq \mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{t+1})$ and set $\mathbb{L}'_{t+1} = \bigl(\mathbb{L}'_t \setminus u\langle U\rangle\bigr) \cup \mathbb{M}_u$. By part 1 of Proposition 8.49, a *lower* bound for the cost locally of the pebbling sequence $\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_t) \rightsquigarrow \mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{t+1})$ in $\mathcal{L}$ is $\max_{i\in\{t,t+1\}}\bigl\{\mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_{i+1}))\bigr\}$. Using Lemma 8.64 and parts 1 and 2 of Proposition 8.49, we can upper-bound the pebbling cost locally inside $cl(u\langle U\rangle)$ in $\mathcal{L}'$ in terms of this local cost in $\mathcal{L}$ by

$$\begin{aligned} \mathit{cost}(u\langle U\rangle \cup \mathbb{M}_u) &\le \mathit{cost}(u\langle U\rangle) + \mathit{cost}(\mathbb{M}_u) \\ &\le 2 \cdot \max_{i\in\{t,t+1\}}\bigl\{\mathit{cost}(\mathsf{proj}_{u\langle U\rangle}(\mathbb{L}_i))\bigr\} \ , \end{aligned} \tag{8.32}$$

which yields invariants 1-4.

Going through the moves in $\mathcal{L} = \{\mathbb{L}_0, \ldots, \mathbb{L}_\tau\}$, this construction yields a labelled pebbling $\mathcal{L}'$ without wasteful implosion moves such that $\mathbb{L}'_{\tau'} \succeq \mathbb{L}_\tau$ and $\mathit{cost}(\mathcal{L}') \le 2 \cdot \mathit{cost}(\mathcal{L})$. □

Thereby, the proof of Lemma 8.31 as outlined in Section 8.5.1 is complete, and Theorem 8.7 follows. We conclude the section by restating the lemma and writing out the proof in full for completeness.

Lemma 8.31 (restated). *Suppose that $\mathcal{L}$ is a complete L-pebbling of a complete binary tree T. Then from $\mathcal{L}$ we can construct a complete L-pebbling $\mathcal{L}^*$ of T without reversals such that $\mathit{cost}(\mathcal{L}^*) = \mathrm{O}(\mathit{cost}(\mathcal{L}))$.*

Proof. Let $\mathcal{L}$ be an arbitrary complete L-pebbling of T. Without loss of generality, we can assume that $\mathcal{L}$ is non-redundant in the sense of Lemma 8.34. By Lemma 8.40, we can also assume that $\mathcal{L}$ contains only simple L-configurations. This sets the stage for applying the technical machinery developed in Sections 8.5.4–8.5.8.

First, using Lemma 8.55 we transform $\mathcal{L}$ into a non-overlapping L-pebbling $\mathcal{L}'$ with $\mathit{cost}(\mathcal{L}') \le \mathit{cost}(\mathcal{L})$. If $\mathcal{L}'$ contains wasteful implosion moves, we then let Lemma 8.66 provide us with a non-wasteful complete L-pebbling $\mathcal{L}''$ such that $\mathit{cost}(\mathcal{L}'') \le 2\cdot\mathit{cost}(\mathcal{L}')$. But for such an L-pebbling, Lemma 8.61 allows us to project away all implosion moves without increasing the pebbling cost, so we finally get a reversal-free complete L-pebbling $\mathcal{L}^*$ of T with $\mathit{cost}(\mathcal{L}^*) \le \mathit{cost}(\mathcal{L}'') \le 2 \cdot \mathit{cost}(\mathcal{L}') \le 2 \cdot \mathit{cost}(\mathcal{L})$. This proves the lemma. □

Chapter 9

Towards Separating Space and Length

The techniques used in Chapter 8 enables us to prove lower bounds on clause space for pebbling contradictions over trees, but as we can see from Lemma 8.6 there is not much hope that the labelled pebble game will give us anything for more general graphs. To improve our bounds to better than logarithmic, we have to do something different.

In this chapter, we therefore devise another pebble game, which makes it possible to prove tight bounds on the clause space of refuting pebbling contradictions over pyramid DAGs. This yields the exponential improvements over Chapter 8 in Theorem 2.3 and Corollary 2.4. This chapter is based on joint work with my advisor Johan Håstad to appear as [63].

9.1 Overview of Improved Lower Bound Proof

We start this chapter by giving a high-level overview of the proof and trying to describe the similarities and differences compared to the approach in Chapter 8. As before, we want to think of a black pebble on v as corresponding to truth of the disjunction $\bigvee_{i=1}^{d} x(v)_i$ of all positive literals over v, or to "truth of v". As we have seen, with this correspondence it is straightforward to translate a pebbling of G using only black pebbles into refutation of the pebbling contradiction Peb_G^d. The only observation needed is that if we have derived the clauses $\bigvee_{i=1}^{d} x(s)_i$ and $\bigvee_{i=1}^{d} x(t)_i$ for the two predecessors s and t of v, then by downloading the axioms saying that truth propagates from s and t to v we can derive $\bigvee_{i=1}^{d} x(v)_i$. The correspondence here is quite close in that the space used by the refutation is at most an additive constant larger than the number of black pebbles used (Proposition 5.10).

When we look at pebblings involving also white pebbles the translation gets slightly more complicated. White pebbles enable us to place black pebbles "in the middle" of the DAG without first having to pebble bottom-up from the sources. For instance, if we white-pebble u and v in Figure 3.2, we can then place a black pebble on their common successor z. Next, the white pebble on, say, v can be eliminated by placing white pebbles on the predecessors s and t, allowing the pebble on v to be removed. A resolution derivation can mimic these pebbling moves by writing all axioms $\overline{x(u)}_i \vee \overline{x(v)}_j \vee \bigvee_{n=1}^{d} x(z)_n$ and $\overline{x(s)}_i \vee \overline{x(r)}_j \vee \bigvee_{n=1}^{d} x(v)_n$ on the blackboard and then using all these clauses to derive $\overline{x(u)}_i \vee \overline{x(s)}_j \vee \overline{x(r)}_l \vee \bigvee_{n=1}^{d} x(z)_n$ for $i, j, l \in [d]$. As it happens, it is possible to translate any black-white pebbling to a refutation in this way (modulo some technical details), but the reduction is not as tight as in the case of a black-pebbles-only pebbling. As we can see from the example above, this naive translation can transform N white pebbles into a set of clauses of size d^N.

The key to our argument, however, is a translation in the other direction. We want to start with a resolution refutation and produce a black-white pebbling and then use the existing lower bound machinery for the black-white pebble game to get a lower bound on clause space. Let us first try to give the intuition behind our translation, and then discuss some technical complications that arise and how we adapt our construction to cope with these problems.

For black pebbles, we can reuse the ideas above for transforming pebblings into refutations. If the clause $\bigvee_{i=1}^{d} x(v)_i$ is implied by the current content of the blackboard, we will let this correspond to a black pebble on v. A white pebble in a pebbling is a “debt” that has to be paid. It is difficult to see how any clause could be a liability in the same way and therefore no set of clauses corresponds naturally to isolated white pebbles. But if we think of white pebbles as assumptions that allow us to place black pebbles higher up in the DAG, it makes sense to say that if the content of the blackboard conditionally implies the clause $\bigvee_{i=1}^{d} x(v)_i$ given that we also assume the set of clauses $\left\{\bigvee_{i=1}^{d} x(w)_i \middle| w \in W\right\}$ for some vertex set W, then this could be interpreted as a black pebble on v and white pebbles on the vertices in W.

Using this correspondence, we can translate sets of clauses into black and white pebbles in a way that fits nicely with the resolution derivations sketched above. To give a concrete example, the clauses

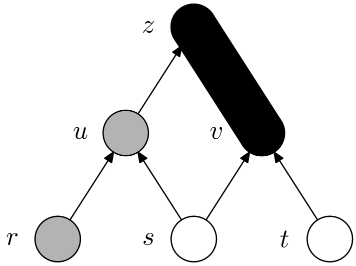
$$\begin{bmatrix} x(u)_1 \vee x(u)_2 \\ \overline{x(s)}_1 \vee \overline{x(t)}_1 \vee x(v)_1 \vee x(v)_2 \\ \overline{x(s)}_1 \vee \overline{x(t)}_2 \vee x(v)_1 \vee x(v)_2 \\ \overline{x(s)}_2 \vee \overline{x(t)}_1 \vee x(v)_1 \vee x(v)_2 \\ \overline{x(s)}_2 \vee \overline{x(t)}_2 \vee x(v)_1 \vee x(v)_2 \end{bmatrix} \tag{9.1}$$

in Figure 3.3(a) correspond to the pebbles in Figure 3.3(b), i.e., black pebbles on u and v and white pebbles on s and t. To see this, note that if we assume $x(s)_1 \vee x(s)_2$ and $x(t)_1 \vee x(t)_2$, this assumption together with the clauses on the blackboard imply $x(v)_1 \vee x(v)_2$, so v is black-pebbled and s and t are white-pebbled in Figure 3.3(b). The vertex u is also black since $x(u)_1 \vee x(u)_2$ certainly is implied by the blackboard.

The problem is that refutations can derive clauses that cannot be translated, at least not naturally, to pebbles in the way indicated above. Some of these clauses

$$\begin{bmatrix} \overline{x(s)_1} \lor \overline{x(t)_1} \lor x(v)_1 \lor x(z)_1 \lor x(z)_2 \\ \overline{x(s)_1} \lor \overline{x(t)_2} \lor x(v)_1 \lor x(z)_1 \lor x(z)_2 \\ \overline{x(s)_2} \lor \overline{x(t)_1} \lor x(v)_1 \lor x(z)_1 \lor x(z)_2 \\ \overline{x(s)_2} \lor \overline{x(t)_2} \lor x(v)_1 \lor x(z)_1 \lor x(z)_2 \end{bmatrix}$$

(a) New set of clauses on blackboard.



(b) Corresponding blobs and pebbles.

Figure 9.1: Interpreting sets of clauses as black blobs and white pebbles.

we can afford to ignore. For example, considering how axiom clauses can be used in derivations it seems reasonable to expect that a derivation never writes an isolated axiom $\overline{x(s)_i} \lor \overline{x(t)_j} \lor x(v)_1 \lor x(v)_2$ on the blackboard. And in fact, if three of the four axioms for v in (9.1) are written on the blackboard but the fourth one $\overline{x(s)_2} \lor \overline{x(t)_2} \lor x(v)_1 \lor x(v)_2$ is missing, we will just discard these three clauses and there will be no pebbles on s, t, or v corresponding to them.

A more dangerous situation is when clauses are derived that are the disjunction of positive literals from different vertices. Such clauses say something about the truth of several vertices but only cost one in terms of clause space. They do not appear to be very useful, but nevertheless we have to model them in some way. To see why, consider the following example. Starting from the blackboard in (9.1), a refutation could add the axioms $\overline{x(u)_i} \lor \overline{x(v)_2} \lor x(z)_1 \lor x(z)_2$ for $i = 1, 2$, derive the clauses $\overline{x(s)_i} \lor \overline{x(t)_j} \lor x(v)_1 \lor x(z)_1 \lor x(z)_2$ for $i, j = 1, 2$, and then erase $x(u)_1 \lor x(u)_2$ to save space, resulting in the blackboard in Figure 9.1(a). This blackboard does not correspond to any pebbles under our tentative translation. However, the clauses can easily be used to derive something that does. For instance, writing down all axioms $\overline{x(u)_i} \lor \overline{x(v)_j} \lor x(z)_1 \lor x(z)_2$ for $i, j = 1, 2$, we get that the truth of s, t, and u implies the truth of z. We have decided to interpret such a set of clauses as a black pebble on z and white pebbles on s, t, and u, but this pebble configuration cannot arise out of nothing in an empty DAG.

Although it is hard to motivate from such a small example, this turns out to be a serious problem. And as we saw in Chapter 8, when we interpreted a set of clauses as in Figure 9.1(a) by placing white pebbles on s and t and a black pebble on v, the result was a pebble game that could only yield lower bounds for trees (intuitively, because black pebbles can slide downwards, which breaks the lower bounds we know for black-white pebbling price), which in turn meant that we could only get logarithmic lower bounds on clause space. We want to do better.

Therefore, we give up the notion that clauses should correspond to pebbles and invent a new "pebble" game, with white pebbles just as before, but with black *blobs* that can cover multiple vertices instead of single-vertex black pebbles. A blob on a vertex set V can be thought of as truth of some vertex $v \in V$. The clauses in Figure 9.1(a) are consequently translated into white pebbles on s and t, as before, and a black blob covering both v and z in Figure 9.1(b).

We use this blob-pebble game to build a lower bound proof as outlined in Section 3.3. First, we establish that for a fairly general class of graphs, any resolution refutation of a pebbling contradiction can be interpreted as a blob-pebbling on the DAG in terms of which this pebbling contradiction is defined.

Theorem 9.1. *Let Peb^d_G denote any pebbling contradiction over a layered DAG G. Then there is a translation from sets of clauses derived from Peb^d_G into sets of black blobs and white pebbles in G such that any resolution refutation π of Peb^d_G corresponds to a blob-pebbling $\mathcal{P}_\pi$ of G under this translation.*

Proof outline. If there are vertex sets B and W with $B \cap W = \emptyset$ and a subset $\mathbb{C}$ of blackboard clauses such that

$$\mathbb{C} \cup \left\{ \bigvee_{i=1}^{d} x(w)_i \middle| w \in W \right\} \vDash \bigvee_{v \in B} \bigvee_{i=1}^{d} x(v)_i \tag{9.2}$$

but this implication does not hold for any subset of B, W, or $\mathbb{C}$, the blackboard *induces* a black blob on B *supported* by white pebbles on W. This *subconfiguration* of pebbles is denoted $[B]\langle W\rangle$. The pebble configuration corresponding to the blackboard is the set $\mathbb{S}$ of all induced subconfigurations.

When an axiom clause $\overline{x(s)}_i \vee \overline{x(t)}_j \vee \bigvee_{l=1}^{d} x(v)_l$ is written on the blackboard, we match this (if needed) by a black blob on v and white pebbles on s and t, i.e., by the subconfiguration $[v]\langle s, t\rangle$. This is an *introduction* move, and it corresponds to the rules for black and white pebble placement in the standard pebble game. If v is a source, we get the subconfiguration $[v]\langle \emptyset \rangle$.

Because of the new axiom clause, there can also appear other new blobs and pebbles. We show that they can all be explained in terms of *mergers* of existing subconfigurations $[B_1]\langle W_1\rangle$ and $[B_2]\langle W_2\rangle$ such that $B_1 \cap W_2 = \emptyset$ and $B_2 \cap W_1 = \{v^*\}$ for some vertex v^* into $\big[(B_1 \cup B_2) \setminus \{v^*\}\big]\big\langle (W_1 \cup W_2) \setminus \{v^*\}\big\rangle$. To see why the merger rule is defined the way it is, note that if the blackboard plus truth of all $w \in W_i$ implies the truth of some $v \in B_i$ for $i = 1, 2$, then certainly the blackboard plus truth of all $w \in (W_1 \cup W_2) \setminus \{v^*\}$ implies the truth of some vertex either in B_1 (if v^* is true) or $B_2 \setminus \{v^*\}$ (if v^* is false).

When new clauses are derived, we expect nothing to happen since these clauses are implied by what is already on the blackboard. It can be the case, though, that clauses are derived that are in some sense "weaker" than what is implied by the blackboard, but if so we can make *inflation* moves that inflate blobs to cover more vertices and/or add white pebbles.

Finally, blobs and pebbles can disappear when clauses are erased from the blackboard. A problem here is that we can get erasures of white pebbles, which is not acceptable in the black-white pebble game. However, by associating white pebbles with black blobs in subconfigurations $[B]\langle W\rangle$, we can allow erasures of white pebbles W as long as the blobs B that they support are erased as well. Thus, one cannot erase individual pebbles but only entire subconfigurations. In this way (sweeping the technical details under the rug for now) we can associate a blob-pebbling $\mathcal{P}_\pi$ with any refutation π. □

In fact, the only property that we need from the layered graphs in Theorem 9.1 is that if w is a vertex with predecessors u and v, then there is no path between the siblings u and v. The theorem holds for any DAG satisfying this condition.

The next step is to design a cost function for black blobs and white pebbles so that the cost of the blob-pebbling $\mathcal{P}_\pi$ in Theorem 9.1 is related to the space of the resolution refutation π. Consider first two special cases. If a clause set induces N disjoint blobs without any supporting white pebbles, it is not hard to prove that the size of this set is at least N. This is clearly tight, so the cost of a single blob can never exceed one. And if $\mathbb{C} \cup \left\{\bigvee_{i=1}^{d} x(w)_i \middle| w \in W\right\}$ implies $\bigvee_{v\in B} \bigvee_{i=1}^{d} x(v)_i$ with the vertex set W chosen minimal so that this implication still holds, it can be shown that $|\mathbb{C}| > (d-1)|W|$ (so here we need $d > 1$). This follows from the fact that a minimally unsatisfiable CNF formula over N variables must contain strictly more than N clauses (Theorem 8.27).

In general, matters will be more complicated. Distinct blobs will not be disjoint, and therefore cannot always all count towards the cost. Also, black blobs and white pebbles from different subconfigurations can intersect in tricky ways. However, it turns out that if we only allow blobs B that are chains (i.e., where all $v \in B$ are ordered topologically), look at the lowest vertex in each blob and count the number of distinct such vertices, and also only charge for white pebbles in $[B]\langle W\rangle$ that are located below the bottom vertex of B, we get the required result. Once we have defined the cost function in this way, the proof that blob-pebbling cost yields a lower bound on clause space is similar to the proof of Theorem 8.29.

Theorem 9.2. *If π is a resolution refutation of a pebbling contradiction of degree $d > 1$, the cost of the associated blob-pebbling $\mathcal{P}_\pi$ is bounded by the space of π by $\mathsf{cost}(\mathcal{P}_\pi) \leq Sp(\pi) + \mathrm{O}(1)$.*

Finally, we need lower bounds on blob-pebbling price. Because of the inflation rule in combination with the peculiar cost function, the blob-pebble game seems to behave somewhat differently from the standard black-white pebble game, and therefore we cannot appeal directly to known lower bounds on black-white pebbling price. Luckily, it so happens that the lower bound construction in [49] that we presented in Section 6.5 can be generalized to the blob-pebble game giving the following theorem.

Theorem 9.3. *Pyramids Π_h have blob-pebbling price $\Theta(h)$.*

Proof outline. This is arguably the technically most complex part of the construction, but let us nevertheless try to briefly outline the proof structure. The key idea (adapted from [49]) is to define a *potential* measure for the set of subconfigurations $\mathbb{S} = \{[B_i]\langle W_i\rangle \mid i = 1, \ldots, m\}$ currently in the graph as an indicator of "how good" this set is. We then prove two facts about this potential, from which the desired lower bound on blob-pebbling price immediately follows:

1. The potential of the current pebble configuration $\mathbb{S}_t$ is upper-bounded, up to a fixed multiplicative constant, by the maximum cost of any configuration $\mathbb{S}_{t'}$, $t' \leq t$.

2. The final pebble configuration $\mathbb{S}_\tau = \big\{[z]\langle\emptyset\rangle\big\}$ consisting of a single black blob on the sink has potential $\Theta(h)$.

More precisely, we let $U\{\succeq j\}$ denote the subset of vertices in U on or above level j (recall that sources are on level 0 and the sink z is on level h) and define $m(U) = \max\big\{j + 2|U\{\succeq j\}| : U\{\succeq j\} \neq \emptyset\big\}$ to be the *measure* of U. We say that the vertex set U *blocks* the subconfiguration $[B]\langle W\rangle$ if $U \cup W$ intersects every path P from a source vertex such that $B \subseteq P$, and that U blocks $\mathbb{S}$ if it blocks every $[B]\langle W\rangle \in \mathbb{S}$. The *potential* of $\mathbb{S}$ is $\mathrm{pot}(\mathbb{S}) = \min\{m(U) : U \text{ blocks } \mathbb{S}\}$.

Fact 2 of the proof now follows easily, since it can be shown that the set U with smallest measure blocking $\mathbb{S}_\tau = \big\{[z]\langle\emptyset\rangle\big\}$ is $U = \{z\}$ with $m(U) = h + 2$.

Fact 1 is the hard part. It is proven by induction. Suppose that U_t blocks $\mathbb{S}_t$ and that $\mathrm{pot}(\mathbb{S}_t) = m(U_t)$. By the inductive hypothesis, we have $\mathrm{pot}(\mathbb{S}_t) \leq C \cdot \max_{t' \leq t}\{\mathit{cost}(\mathbb{S}_{t'})\}$. We want to show $\mathrm{pot}(\mathbb{S}_{t+1}) \leq C \cdot \max_{t' \leq t+1}\{\mathit{cost}(\mathbb{S}_{t'})\}$, which clearly holds if

$$\mathrm{pot}(\mathbb{S}_{t+1}) \leq \max\{\mathrm{pot}(\mathbb{S}_t), C \cdot \mathit{cost}(\mathbb{S}_{t+1})\} \ . \tag{9.3}$$

To establish this inequality, we note that if the pebbling move at time $t + 1$ is an inflation, $\mathbb{S}_{t+1}$ is blocked by U_t. Hence, $\mathrm{pot}(\mathbb{S}_{t+1}) \leq m(U_t) = \mathrm{pot}(\mathbb{S}_t)$ in this case. In the same way it can be verified that if U_t blocks two subconfigurations being merged, it must also block the result of the merger. And if we make an introduction move on a non-source vertex v, the white pebbles on the predecessors of v block the black pebble on v no matter what U_t looks like.

Thus, the potential can only increase when an introduction of $[v]\langle\emptyset\rangle$ is performed on a source v. It turns out that what we need to prove (9.3) in this case is that pyramid graphs have the following property: *There exists a constant C' such that for any configuration $\mathbb{S}$ there is a blocking vertex set U with* $\mathrm{pot}(\mathbb{S}) = m(U)$ *and* $|U| \leq C' \cdot \mathit{cost}(\mathbb{S})$.

This far the construction closely parallels that in Section 6.5, but showing that we can choose blocking sets that achieve the minimum measure and at the same time have limited cardinality requires new tools, as well as using the proof in Section 6.5 as a subroutine. □

We remark that the proof of Theorem 9.3 applies in (almost) the same generality as in [49]. It works for all layered DAGs that are also "spreading" in the sense that (loosely speaking) for every vertex v on any level L and every $K \leq L$, there are at least $K+1$ vertices located exactly K levels below v from which v is reachable. This class of graphs includes among others complete binary trees and pyramid graphs.

It is an intriguing open question to determine the exact relation between the blob-pebble game and the black-white pebble game. On the one hand, to prove Theorem 9.3 we use additional techniques and get worse constants compared to the construction in [49]. On the other hand, we do not know of a single example where the possibility to use blobs reduces the cost of the cheapest pebbling.

Returning to our main path of reasoning and putting all of this together, we can now prove the main theorem of this chapter.

Theorem 2.3 (restated). *Let $Peb^d_{\Pi_h}$ denote the pebbling contradiction of degree $d > 1$ defined over the pyramid graph of height h. Then the clause space of refuting $Peb^d_{\Pi_h}$ by resolution is $Sp(Peb^d_{\Pi_h} \vdash 0) = \Theta(h)$.*

Proof. The upper bound $Sp(Peb^d_{\Pi_h} \vdash 0) = \mathrm{O}(h)$ is easy. A pyramid of height h can be pebbled with $h + \mathrm{O}(1)$ black pebbles, and as was noted above a refutation can mimic such a pebbling in constant extra clause space (independent of d).

As before, the interesting part is the lower bound. Let π be any resolution refutation of $Peb^d_{\Pi_h}$ and consider the associated blob-pebbling $\mathcal{P}_\pi$ provided by Theorem 9.1. On the one hand, we know that $\mathit{cost}(\mathcal{P}_\pi) = \mathrm{O}(Sp(\pi))$ by Theorem 9.2, provided that $d > 1$. On the other hand, Theorem 9.3 tells us that the cost of any blob-pebbling of Π_h is $\Omega(h)$, so in particular we must have $\mathit{cost}(\mathcal{P}_\pi) = \Omega(h)$. Combining these two bounds on $\mathit{cost}(\mathcal{P}_\pi)$, we see that $Sp(\pi) = \Omega(h)$. □

Recall that the pebbling contradiction Peb^d_G is a $(2+d)$-CNF formula and that for constant d the size of the formula is linear in the number of vertices of G. Hence, for pyramid graphs Π_h the corresponding pebbling contradictions $Peb^d_{\Pi_h}$ have size quadratic in the height h. Also, when d is fixed the upper bounds in Proposition 5.7 become $L(Peb^d_G \vdash 0) = \mathrm{O}(n)$ and $W(Peb^d_G \vdash 0) = \mathrm{O}(1)$. Corollary 2.4 now follows if we set $F_n = Peb^d_{\Pi_h}$ for $d = k - 2$ and $h = \lfloor \sqrt{n} \rfloor$ and use Theorem 2.3.

Corollary 2.4 (restated). *For every $k \geq 4$, there is a family of k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ that can be refuted in length $L(F_n \vdash 0) = \mathrm{O}(n)$ and width $W(F_n \vdash 0) = \mathrm{O}(1)$ but require clause space $Sp(F_n \vdash 0) = \Theta(\sqrt{n})$.*

9.2 The Blob-Pebble Game

The rest of this chapter is devoted to proving Theorem 2.3. In this section, we present the modified pebble game that we will use to study the clause space of resolution refutations of pebbling contradictions. For our pebble game to work, we require of the graphs under study that they have the following property.

Property 9.4 (Sibling non-reachability). We say that a DAG G has the *Sibling non-reachability property* if for all vertices u and v that are siblings in G, it holds that $u \notin G^v_\triangle$ and $v \notin G^u_\triangle$, i.e., the siblings are not reachable from one another.

Phrased differently, Property 9.4 asserts that siblings are non-comparable.

A sufficient condition for Property 9.4 to hold is that if v is reachable from u, then all paths $P : u \rightsquigarrow v$ have the same length. This holds for instance for the class of layered graphs in Definition 6.2, and it is also easy to see directly that layered graphs possess Property 9.4.

In the rest of this chapter, we will concentrate on DAGs having certain structural properties. The next definition is so that we will not have to repeat these properties over and over again.

Definition 9.5 (Blob-pebblable DAG). A *blob-pebblable DAG* is a DAG that has a unique sink, which we will alway denote z, that has vertex indegree 2 for all non-sources, and that satisfies the Sibling non-reachability property 9.4.

9.2.1 Description of the Blob-Pebble Game and Formal Definition

As before, we alter the rule for white pebble removal so that a white pebble can be removed from a vertex when a black pebble is placed on that same vertex. This will make the correspondence between pebblings and resolution derivations much more natural. Clearly, this is only a minor adjustment, and it is easy to prove formally that it does not really change anything.

Also, we still have the problem that in general, a resolution refutation has no reason a priori to follow our pebble game intuition. Since pebbles are induced by clauses, if at some derivation step the refutation chooses to erase "the wrong clause" from the point of view of the induced pebble configuration, this can lead to pebbles just disappearing. Whatever our translation from clauses to pebbles is, a resolution proof that suddenly out of spite erases practically all clauses must surely lead to practically all pebbles disappearing, at least if we want to maintain a correspondence between clause space and pebbling cost. This is all in order for black pebbles, but if we allow uncontrolled removal of white pebbles we cannot hope for any nontrivial lower bounds on pebbling price (just white-pebble the two predecessors of the sink, then black-pebble the sink itself and finally remove the white pebbles).

We will keep our solution to this problem from Chapter 8, namely, to keep track of exactly which white pebbles have been used to get a black pebble on a vertex. We do the necessary bookkeeping by defining *subconfigurations* of pebble configurations, each subconfiguration consisting of black pebble together with all the white pebbles this black pebble depends on, and require that if any pebble in a subconfiguration is removed, then all other pebbles in this subconfiguration must be removed as well.

Another problem is that resolution derivation steps can be made that appear intuitively bad given that we know that the end goal is to derive the empty clause, but where it seems hard to nail down formally wherein this supposed badness lies. To analyze such apparently non-optimal derivation steps, we introduce an *inflation* rule in which a black pebble can be inflated to a *blob* covering multiple vertices. The way to think of this is that a black pebble on a vertex v corresponds to derived truth of v, whereas for a blob pebble on V we only know that some vertex $v \in V$ is true, but not which one. For reasons that will perhaps become clearer in Section 9.5, in is natural to consider blobs that are chains (Definition 6.5).

We now present the formal definition of the concept used to "label" each black blob pebble with the set of white pebbles (if any) this black pebble is dependent on. The intended meaning of the notation $[B]\langle W \rangle$ is a black blob on B together with the white pebbles W with the help of which we have been able to place the black blob on B. These "associated" or "supporting" white pebbles can be located on any vertex $w \notin B$ that can be visited by a source path P to $\mathrm{top}(B)$ agreeing with B. Formally, the *legal pebble positions* with respect to a chain B with $b = \mathrm{bot}(B)$ is the set of vertices

$$lpp(B) = G_{\Delta}^{\not{b}} \cup \left(\bigcup \mathfrak{P}_{\mathrm{in}}(B) \setminus B \right) = \bigcup \mathfrak{P}_{\mathrm{via}}(B) \setminus B \ . \tag{9.4}$$

We refer to the structure $[B]\langle W \rangle$ grouping together a black blob B and its associated white pebbles W as a *blob subconfiguration*, or just *subconfiguration* for short.

Definition 9.6 (Blob subconfiguration). For sets of vertices B, W in a blob-pebblable DAG G, $[B]\langle W \rangle$ is a *blob subconfiguration* if $B \neq \emptyset$ is a chain and $W \subseteq lpp(B)$. We refer to B as a (single) black *blob* and to W as (a number of different) white pebbles *supporting* B. We also say that B is *dependent* on W. If $W = \emptyset$, B is *independent*. Blobs B with $|B| = 1$ are said to be *atomic*.

A set of blob subconfigurations $\mathbb{S} = \bigl\{ [B_i]\langle W_i \rangle \mid i = 1, \ldots, m \bigr\}$ together constitute a *blob-pebbling configuration*.

Note in particular that it always holds that $B \cap W = \emptyset$ for a blob subconfiguration $[B]\langle W \rangle$.

Since the definition of the game we will play with these blobs and pebbles is somewhat involved, let us first try to give an intuitive description.

- There is one single rule corresponding to the two rules 1 and 3 for black and white pebble placement in the black-white pebble game of Definition 5.1. This *introduction* rule says that we can place a black pebble on a vertex v together with white pebbles on its predecessors (unless v is a source, in which case no white pebbles are needed).

- The analogy for rule 2 for black pebble removal in Definition 5.1 is a rule for "shrinking" black blobs. A vertex v in a blob can be eliminated by *merging* two blob subconfigurations, provided that there is both a black blob and a white pebble on v, and provided that the two black blobs involved in this *merger* do not intersect the supporting white pebbles of one another in any other vertex than v. Removing black pebbles in the black-white pebble game corresponds to shrinking atomic black blobs.

- A black blob can be *inflated* to cover more vertices, as long as it does not collide with its own supporting white vertices. Also, new supporting white pebbles can be added at an inflation move. There is no analogy of this move in the usual black-white pebble game.

- The rule 4 for white pebble removal also corresponds to merging in the blob-pebble game, since the white pebble used in the merger is eliminated as well. In addition, however, a white pebble on w can also disappear if its black blob B changes so that w no longer can be visited on a path via B (i.e., if w is no longer a legal pebble position with respect to B).
- Other than that, individual white pebbles, and individual black vertices covered by blobs, can never just disappear. If we want to remove a white pebble $w \in W$ or parts of a black blob B, we can do so only by *erasing* the whole blob subconfiguration $[B]\langle W\rangle$.

The formal definition follows. See Figure 9.2 for some examples of blob-pebbling moves.

Definition 9.7 (Blob-pebble game). For a blob-pebblable DAG G and blob-pebbling configurations $\mathbb{S}_0$ and $\mathbb{S}_\tau$ on G, a *blob-pebbling* from $\mathbb{S}_0$ to $\mathbb{S}_\tau$ in G is a sequence $\mathcal{P} = \{\mathbb{S}_0, \ldots, \mathbb{S}_\tau\}$ of configurations such that for all $t \in [\tau]$, $\mathbb{S}_t$ is obtained from $\mathbb{S}_{t-1}$ by one of the following rules:

Introduction $\mathbb{S}_t = \mathbb{S}_{t-1} \cup \{[v]\langle \mathit{pred}(v)\rangle\}$.

Merger $\mathbb{S}_t = \mathbb{S}_{t-1} \cup \{[B]\langle W\rangle\}$ if there are $[B_1]\langle W_1\rangle, [B_2]\langle W_2\rangle \in \mathbb{S}_{t-1}$ such that

1. $B_1 \cup B_2$ is (totally) ordered,
2. $B_1 \cap W_2 = \emptyset$,
3. $|B_2 \cap W_1| = 1$; let v^* denote this unique element in $B_2 \cap W_1$,
4. $B = (B_1 \cup B_2) \setminus \{v^*\}$, and
5. $W = \big((W_1 \cup W_2) \setminus \{v^*\}\big) \cap \mathit{lpp}(B)$,

We write $[B]\langle W\rangle = \mathsf{merge}([B_1]\langle W_1\rangle, [B_2]\langle W_2\rangle)$ and refer to this as a *merger on v^**.

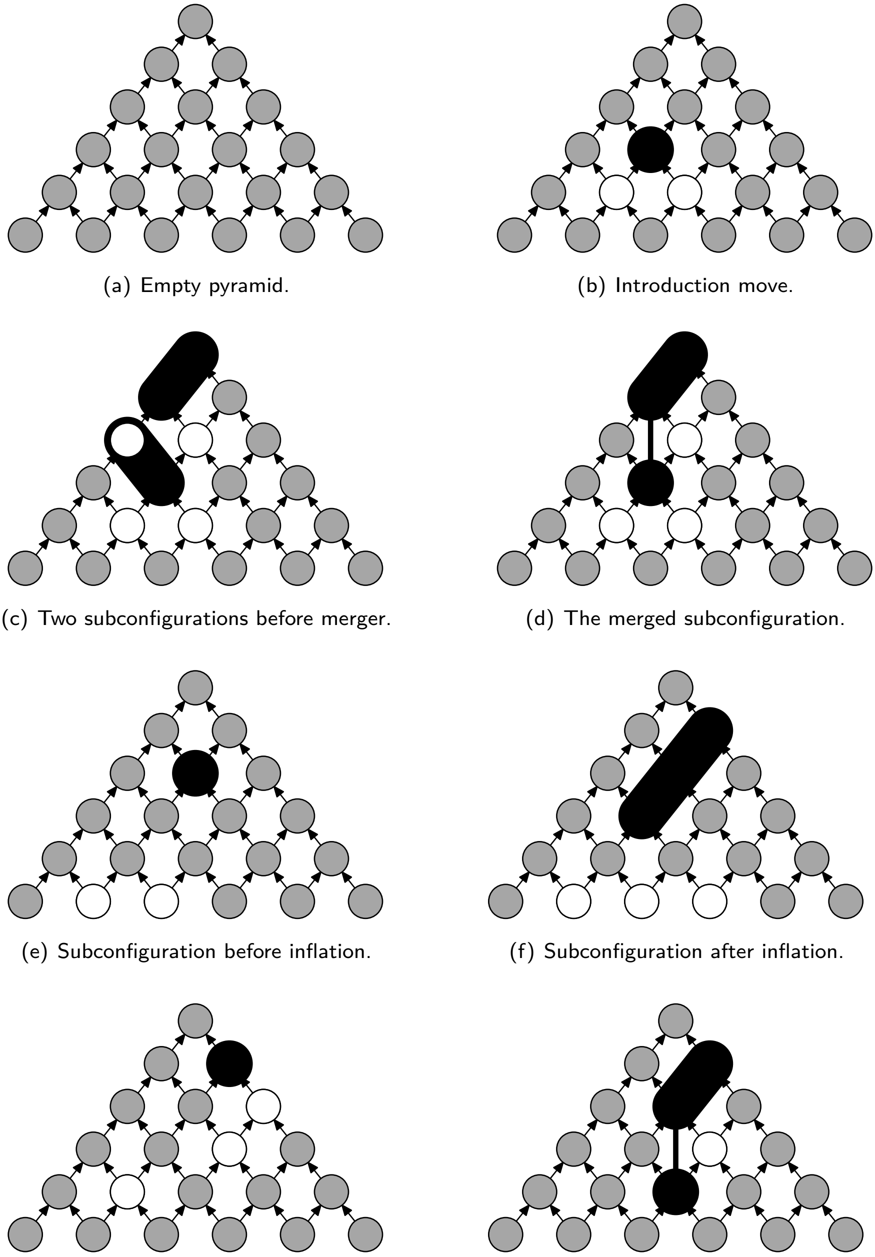
Inflation $\mathbb{S}_t = \mathbb{S}_{t-1} \cup \{[B]\langle W\rangle\}$ if there is a $[B']\langle W'\rangle \in \mathbb{S}_{t-1}$ such that

1. $B \supseteq B'$,
2. $B \cap W' = \emptyset$, and
3. $W \supseteq W' \cap \mathit{lpp}(B)$.

We say that $[B]\langle W\rangle$ is derived from $[B']\langle W'\rangle$ by inflation or that $[B']\langle W'\rangle$ is *inflated* to yield $[B]\langle W\rangle$.

Erasure $\mathbb{S}_t = \mathbb{S}_{t-1} \setminus \{[B]\langle W\rangle\}$ for $[B]\langle W\rangle \in \mathbb{S}_{t-1}$.

The blob-pebbling $\mathcal{P}$ is *unconditional* if $\mathbb{S}_0 = \emptyset$ and *conditional* otherwise. A *complete blob-pebbling* of G is an unconditional pebbling $\mathcal{P}$ ending in $\mathbb{S}_\tau = \{[z]\langle\emptyset\rangle\}$ for z the unique sink of G.



(a) Empty pyramid.

(b) Introduction move.

(c) Two subconfigurations before merger.

(d) The merged subconfiguration.

(e) Subconfiguration before inflation.

(f) Subconfiguration after inflation.

(g) Another subconfiguration before inflation.

(h) After inflation with vanished white pebbles.

Figure 9.2: Examples of moves in the blob-pebble game.

9.2.2 Blob-Pebbling Price

We have not yet defined what the price of a blob-pebbling is. The reason is that it is not a priori clear what the "correct" definition of blob-pebbling price should be.

It should be pointed out that the blob-pebble game has no obvious intrinsic value—its function is to serve as a tool to prove lower bounds on the resolution refutation space of pebbling contradictions. The intended structure of our lower bound proof for resolution space is that we want to look at resolution refutations of pebbling contradictions, interpret them in terms of blob-pebblings on the underlying graphs, and then translate lower bounds on the price of these blob-pebblings into lower bounds on the size of the corresponding clause configurations. Therefore, we have two requirements for the blob-pebbling price $\mathsf{Blob\text{-}Peb}(G)$:

1. It should be sufficiently high to enable us to prove good lower bounds on $\mathsf{Blob\text{-}Peb}(G)$, preferably by relating it to the standard black-white pebbling price $\mathsf{BW\text{-}Peb}(G)$.
2. It should also be sufficiently low, so that lower bounds on $\mathsf{Blob\text{-}Peb}(G)$ translate back into lower bounds on the size of the clause configurations.

So when defining pebbling price in Definition 9.8 below, we also have to have in mind the coming Definition 9.9 saying how we will interpret clauses in terms of blobs and pebbles and that these two definitions together should make it possible for us to lower-bound clause set size in terms of pebbling cost.

For black pebbles, we could try to charge 1 for each distinct blob. But this will not work, since then the second requirement above fails. For the translation of clauses to blobs and pebbles sketched in Section 9.1 it is possible to construct clause configurations that correspond to an exponential number of distinct black blobs measured in the clause set size. The other natural extreme seems to be to charge only for mutually disjoint black blobs. But this is far too generous, and the first requirement above fails. To get a trivial example of this, take any ordinary black pebbling of G and translate in into an (atomic) blob-pebbling, but then change it so that each black pebble $[v]$ is immediately inflated to $[\{v, z\}]$ after each introduction move. It is straightforward to verify that this would yield a pebbling of G in constant cost.

For white pebbles, the first idea might be to charge 1 for every white-pebbled vertex, just as in the standard pebble game. On closer inspection, though, this seems to be not quite what we need.

The definition presented below turns out to give us both of the desired properties above, and allows us to prove an optimal bound. Namely, we define blob-pebbling price so as to charge 1 for *each distinct bottom vertex* among the black blobs, and so as to charge for the subset of supporting white pebbles $W \cap G^b_\vartriangle$ in a subconfiguration $[B]\langle W\rangle$ that are *located below the bottom vertex* $\mathrm{bot}(B)$ of the black blob B. Multiple distinct blobs with the same bottom vertex come for free, however, and any supporting white pebbles above the bottom vertex of its own blob are also free, although we still have to keep track of them.

Definition 9.8 (Blob-pebbling price). For a subconfiguration $[B]\langle W\rangle$, we say that $\mathcal{B}([B]\langle W\rangle) = \{\mathrm{bot}(B)\}$ is the *chargeable black vertex* and that $\mathcal{W}^{\Delta}([B]\langle W\rangle) = W \cap G_{\Delta}^{\mathrm{bot}(B)}$ are the *chargeable white vertices*. The *chargeable vertices* of the subconfiguration $[B]\langle W\rangle$ are all vertices in the union $\mathcal{B}([B]\langle W\rangle) \cup \mathcal{W}^{\Delta}([B]\langle W\rangle)$. This definition is extended to blob-pebbling configurations $\mathbb{S}$ in the natural way by letting

$$\mathcal{B}(\mathbb{S}) = \bigcup_{[B]\langle W\rangle \in \mathbb{S}} \mathcal{B}([B]\langle W\rangle) = \bigl\{\mathrm{bot}(B) \mid [B]\langle W\rangle \in \mathbb{S}\bigr\}$$

and

$$\mathcal{W}^{\Delta}(\mathbb{S}) = \bigcup_{[B]\langle W\rangle \in \mathbb{S}} \mathcal{W}^{\Delta}([B]\langle W\rangle) = \bigcup_{[B]\langle W\rangle \in \mathbb{S}} \Bigl(W \cap G_{\Delta}^{\mathrm{bot}(B)}\Bigr) \enspace .$$

The cost of a blob-pebbling configuration $\mathbb{S}$ is $\mathsf{cost}(\mathbb{S}) = \bigl|\mathcal{B}(\mathbb{S}) \cup \mathcal{W}^{\Delta}(\mathbb{S})\bigr|$, and the cost of a blob-pebbling $\mathcal{P} = \bigl\{\mathbb{S}_0, \ldots, \mathbb{S}_\tau\bigr\}$ is $\mathsf{cost}(\mathcal{P}) = \max_{t \in [\tau]}\bigl\{\mathsf{cost}(\mathbb{S}_t)\bigr\}$.

The *blob-pebbling price* of $[B]\langle W\rangle$, denoted $\mathsf{Blob\text{-}Peb}([B]\langle W\rangle)$, is the minimal cost of any unconditional blob-pebbling $\mathcal{P} = \{\mathbb{S}_0, \ldots, \mathbb{S}_\tau\}$ such that $\mathbb{S}_\tau = \bigl\{[B]\langle W\rangle\bigr\}$. The blob-pebbling price of a DAG G is $\mathsf{Blob\text{-}Peb}(G) = \mathsf{Blob\text{-}Peb}([z]\langle\emptyset\rangle)$, i.e., the minimal cost of any complete blob-pebbling of G.

We will also write $\mathcal{W}(\mathbb{S})$ to denote the set of all white-pebbled vertices in $\mathbb{S}$, including non-chargeable ones.

9.3 Resolution Derivations Induce Blob-Pebblings

For simplicity, in this section, as well as in the next one, we will write $v_1, \ldots, v_d$ instead of $x(v)_1, \ldots, x(v)_d$ for the d variables associated with v in a dth degree pebbling contradiction. That is, in Sections 9.3 and 9.4 lower-case letters with subscripts will denote only variables in propositional logic and nothing else.

As in the previous chapter, we find it more natural to ignore the target axioms $\overline{z}_1, \ldots, \overline{z}_d$ and focus on resolution derivations of $\bigvee_{l=1}^{d} z_l$ from the rest of the formula rather than resolution refutations of all of Peb_G^d. Recall that we write $*Peb_G^d = Peb_G^d \setminus \bigl\{\overline{z}_1, \ldots, \overline{z}_d\bigr\}$ to denote the pebbling formula over G with the target axioms in the pebbling contradiction removed (Definition 8.8). Then we know from Lemma 8.9 that for any DAG G with sink z, it holds that $Sp(Peb_G^d \vdash 0) = Sp(*Peb_G^d \vdash \bigvee_{l=1}^{d} z_l)$. In view of this, from now on we will only consider resolution derivations from $*Peb_G^d$ and try to convert clause configurations in such derivations into sets of blob subconfigurations.

Also as in the previous chapter, to avoid cluttering the notation with an excessive amount of brackets, we will sometimes use sloppy notation for sets. We will allow ourselves to omit curly brackets around singleton sets when this is clear from context, writing for instance $V \cup v$ instead of $V \cup \{v\}$ and $[B \cup b]\langle W \cup w\rangle$ instead of $[B \cup \{b\}]\langle W \cup \{w\}\rangle$. Also, we will sometimes omit the curly brackets around sets of vertices in black blobs and write, for instance, $[u, v]$ instead of $[\{u, v\}]$.

9.3.1 Definition of Induced Configurations and Theorem Statement

Recall from Definition 8.8 that if r is a non-source with predecessors $\mathit{pred}(r) = \{p, q\}$, we say that the *axioms for r in *Peb^d_G* is the set

$$Ax^d(r) = \left\{\overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} r_l \mid i, j \in [d]\right\} \tag{9.5}$$

and if r is a source, we define $Ax^d(r) = \left\{\bigvee_{i=1}^{d} r_i\right\}$. For V a set of vertices in G, we let $Ax^d(V) = \left\{Ax^d(v) \mid v \in V\right\}$. Also, we will again use the shorthand notation

$$\mathbb{B}(V) = \left\{\bigvee_{i=1}^{d} v_i \mid v \in V\right\} \tag{9.6}$$

and

$$All^+(V) = \bigvee_{v \in V} \bigvee_{i=1}^{d} v_i \tag{9.7}$$

from Definition 8.12. As before, we say that a set of clauses $\mathbb{C}$ implies a clause D *minimally* if $\mathbb{C} \vDash D$ but for all $\mathbb{C}' \subsetneqq \mathbb{C}$ it holds that $\mathbb{C}' \nvDash D$. We say that $\mathbb{C}$ implies a clause D *maximally* if $\mathbb{C} \vDash D$ but for all $D' \subsetneqq D$ it holds that $\mathbb{C}' \nvDash D'$. To define our translation of clauses into blob subconfigurations, we use implications that are in a sense both minimal and maximal. We remind the reader that the vertex set $\mathit{lpp}(B)$ of legal pebble positions for white pebbles with respect to the chain B was defined in Equation (9.4).

Definition 9.9 (Induced blob subconfiguration). Let G be a blob-pebblable DAG and $\mathbb{C}$ a clause configuration derived from *Peb^d_G. Then $\mathbb{C}$ induces the blob subconfiguration $[B]\langle W\rangle$ if there is a clause set $\mathbb{C}_B \subseteq \mathbb{C}$ and a vertex set $S \subseteq G \setminus B$ with $W = S \cap \mathit{lpp}(B)$ such that

$$\mathbb{C}_B \cup \mathbb{B}(S) \vDash All^+(B) \tag{9.8a}$$

but for which it holds for all strict subsets $\mathbb{C}'_B \subsetneqq \mathbb{C}_B$, $S' \subsetneqq S$ and $B' \subsetneqq B$ that

$$\mathbb{C}'_B \cup \mathbb{B}(S) \nvDash All^+(B) \ , \tag{9.8b}$$

$$\mathbb{C}_B \cup \mathbb{B}(S') \nvDash All^+(B) \ , \text{ and} \tag{9.8c}$$

$$\mathbb{C}_B \cup \mathbb{B}(S) \nvDash All^+(B') \ . \tag{9.8d}$$

We write $\mathbb{S}(\mathbb{C})$ to denote the set of all blob subconfigurations induced by $\mathbb{C}$.

To save space, when all conditions (9.8a)–(9.8d) hold, we write

$$\mathbb{C}_B \cup \mathbb{B}(S) \rhd All^+(B) \tag{9.9}$$

and refer to this as *precise implication* or say that the clause set $\mathbb{C}_B \cup \mathbb{B}(S)$ implies the clause $All^+(B)$ *precisely*. Also, we say that the precise implication $\mathbb{C}_B \cup \mathbb{B}(S) \rhd All^+(B)$ *witnesses* the induced blob subconfiguration $[B]\langle W\rangle$.

In the following, we will use the definition of precise implication $\rhd$ also for clauses $All^{+}(V)$ where the vertex set V is not a chain.

Let us see that this definition agrees with the intuition presented in Section 9.1. An atomic black pebble on a single vertex v corresponds, as promised, to the fact that $\bigvee_{i=1}^{d} v_i$ is implied by the current set of clauses. A black blob on V without supporting white pebbles is induced precisely when the disjunction $All^{+}(V) = \bigvee_{v \in V} \bigvee_{i=1}^{d} v_i$ of the corresponding clauses follow from the clauses in memory, but no disjunction over a strict subset of vertices $V' \subsetneqq V$ is implied. Finally, the supporting white pebbles just indicate that if we indeed had the information corresponding to black pebbles on these vertices, the clause corresponding to the supported black blob could be derived. Remember that our cost measure does not take into account the size of blobs. This is natural since we are interested in clause space, and since large blobs, in an intuitive sense, corresponds to large (i.e., wide) clauses rather than many clauses.

The main result of this section is as follows.

Theorem 9.10. *Let* $\pi = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ *be a resolution derivation of* $\bigvee_{i=1}^{d} z_i$ *from* $*Peb_G^d$ *for a blob-pebblable DAG* G*. Then the induced blob-pebbling configurations* $\{\mathbb{S}(\mathbb{C}_0), \ldots, \mathbb{S}(\mathbb{C}_\tau)\}$ *form the "backbone" of a complete blob-pebbling* $\mathcal{P}$ *of* G *in the sense that*

- $\mathbb{S}(\mathbb{C}_0) = \emptyset$,
- $\mathbb{S}(\mathbb{C}_\tau) = \{[z]\langle\emptyset\rangle\}$, *and*
- *for every* $t \in [\tau]$*, the transition* $\mathbb{S}(\mathbb{C}_{t-1}) \rightsquigarrow \mathbb{S}(\mathbb{C}_t)$ *can be accomplished in accordance with the blob-pebbling rules in cost* $\max\{\mathsf{cost}(\mathbb{S}(\mathbb{C}_{t-1})), \mathsf{cost}(\mathbb{S}(\mathbb{C}_t))\} + \mathrm{O}(1)$.

In particular, to any resolution derivation $\pi : *Peb_G^d \vdash \bigvee_{i=1}^{d} z_i$ *we can associate a complete blob-pebbling* $\mathcal{P}_\pi$ *of* G *such that* $\mathsf{cost}(\mathcal{P}_\pi) \leq \max_{\mathbb{C} \in \pi}\{\mathsf{cost}(\mathbb{S}(\mathbb{C}))\} + \mathrm{O}(1)$.

We prove the theorem by forward induction over the derivation π. By the pebbling rules in Definition 9.7, any subconfiguration $[B]\langle W\rangle$ may be erased freely at any time. Consequently, we need not worry about subconfigurations disappearing during the transition from $\mathbb{C}_{t-1}$ to $\mathbb{C}_t$. What we do need to check, though, is that no subconfiguration $[B]\langle W\rangle$ appears inexplicably in $\mathbb{S}(\mathbb{C}_t)$ as a result of a derivation step $\mathbb{C}_{t-1} \rightsquigarrow \mathbb{C}_t$, but that we can always derive any $[B]\langle W\rangle \in \mathbb{S}(\mathbb{C}_t) \setminus \mathbb{S}(\mathbb{C}_{t-1})$ from $\mathbb{S}(\mathbb{C}_{t-1})$ by the blob-pebbling rules. Also, when several pebbling moves are needed to get from $\mathbb{S}(\mathbb{C}_t)$ to $\mathbb{S}(\mathbb{C}_{t-1})$, we need to check that these intermediate moves do not affect the pebbling cost by more than an additive constant.

The proof boils down to a case analysis of the different possibilities for the derivation step $\mathbb{C}_{t-1} \rightsquigarrow \mathbb{C}_t$. Since the analysis is quite lengthy, we divide it into subsections. But first of all we need some technical lemmas.

9.3.2 Some Technical Lemmas

The next three lemmas are not hard, but will prove quite useful. The first two lemmas were proven already in Chapter 8, but we repeat them here for reference.

Lemma 8.17 (restated). *Suppose that C, D are clauses and that $\mathbb{C}$ is a set of clauses. Then $\mathbb{C} \cup \{C\} \vDash D$ if and only if $\mathbb{C} \vDash \overline{a} \lor D$ for all $a \in \mathit{Lit}(C)$.*

Lemma 8.26 (restated). *Let $\mathbb{C}$ be a set of clauses and D a clause such that $\mathbb{C} \vDash D$ minimally and $a \in \mathit{Lit}(\mathbb{C})$ but $\overline{a} \notin \mathit{Lit}(\mathbb{C})$. Then $a \in \mathit{Lit}(D)$.*

Lemma 9.11. *Suppose that $\mathbb{C} \vDash D$ minimally. Then no literal from D can occur negated in $\mathbb{C}$, i.e., it holds that $\{\overline{a} \mid a \in \mathit{Lit}(D)\} \cap \mathit{Lit}(\mathbb{C}) = \emptyset$.*

Proof. Suppose not. Let $\mathbb{C}_1 = \{C \in \mathbb{C} \mid \exists a \text{ such that } \overline{a} \in \mathit{Lit}(C) \text{ and } a \in \mathit{Lit}(D)\}$ and $\mathbb{C}_2 = \mathbb{C} \setminus \mathbb{C}_1$. Since $\mathbb{C}_2 \nvDash D$, there exists a truth value assignment α such that $\alpha(\mathbb{C}_2) = 1$ and $\alpha(D) = 0$. But then we must have $\alpha(\mathbb{C}_1) = 1$, since every $C \in \mathbb{C}_1$ contains a negated literal $\overline{a}$ from D, and these literals are all set to true by α. Contradiction. □

We also need the following key technical lemma connecting implication with inflation moves (which can be seen to be a parallel of Lemma 8.16).

Lemma 9.12. *Let $\mathbb{C}$ be a clause set derived from $^*\mathit{Peb}^d_G$. Suppose that B is a chain and that $S \subseteq G \setminus B$ is a vertex set such that $\mathbb{C} \cup \mathbb{B}(S) \vDash \mathit{All}^+(B)$ and let $W = S \cap \mathit{lpp}(B)$. Then the blob subconfiguration $[B]\langle W\rangle$ is derivable by inflation from some $[B']\langle W'\rangle \in \mathbb{S}(\mathbb{C})$.*

Proof. Pick $\mathbb{C}' \subseteq \mathbb{C}$, $S' \subseteq S$ and $B' \subseteq B$ minimal such that $\mathbb{C}' \cup \mathbb{B}(S') \vDash \mathit{All}^+(B')$. Then $\mathbb{C}' \cup \mathbb{B}(S') \rhd \mathit{All}^+(B')$ by definition. Note, furthermore, that $B' \neq \emptyset$ since the clause set on the left-hand side must be non-contradictory. Also, $\mathbb{C}' \neq \emptyset$ since $B' \cap S' \subseteq B \cap S = \emptyset$, so by Lemma 8.26 it cannot be that $\mathbb{B}(S') \vDash \mathit{All}^+(B')$. This means that $\mathbb{C}$ induces $[B']\langle W'\rangle$ for $W' = S' \cap \mathit{lpp}(B')$. We claim that $[B']\langle W'\rangle$ can be inflated to $[B]\langle W\rangle$, from which the lemma follows.

To verify this claim, note that first two conditions $B' \subseteq B$ and $B \cap W' \subseteq B \cap S = \emptyset$ for inflation moves in Definition 9.7 clearly hold by construction. As to the third condition, we have

$$W' \cap \mathit{lpp}(B) = \big(S' \cap \mathit{lpp}(B')\big) \cap \mathit{lpp}(B) \subseteq S \cap \mathit{lpp}(B) = W$$

which proves the claim. □

We now start the case analysis in the proof of Theorem 9.10 for the different possible derivation steps in a resolution derivation.

9.3.3 Erasure

Suppose that $\mathbb{C}_t = \mathbb{C}_{t-1} \setminus \{C\}$ for $C \in \mathbb{C}_{t-1}$. It is easy to see that the only possible outcome of erasing clauses is that blob subconfigurations disappear. We note for future reference that this implies that the blob-pebbling cost decreases monotonically when going from $\mathbb{S}(\mathbb{C}_{t-1})$ to $\mathbb{S}(\mathbb{C}_t)$.

9.3.4 Inference

Suppose that $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C\}$ for some clause C derived from $\mathbb{C}_{t-1}$. No blob subconfigurations can disappear at an inference move since $\mathbb{C}_{t-1} \subseteq \mathbb{C}_t$. Suppose that $[B]\langle W\rangle$ is a new subconfiguration at time t arising from $\mathbb{C}_B \subseteq \mathbb{C}_{t-1}$ and $S \subseteq G \setminus B$ such that $W = S \cap \mathit{lpp}(B)$ and $\mathbb{C}_B \cup \{C\} \cup \mathbb{B}(S) \rhd \mathit{All}^+(B)$. Since C is derived from $\mathbb{C}_{t-1}$, we have $\mathbb{C}_{t-1} \vDash C$. Thus it holds that $\mathbb{C}_{t-1} \cup \mathbb{B}(S) \vDash \mathit{All}^+(B)$ and Lemma 9.12 tells us that $[B]\langle W\rangle$ is derivable by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$.

Since no subconfiguration disappears, the pebbling cost increases monotonically when going from $\mathbb{S}(\mathbb{C}_{t-1})$ to $\mathbb{S}(\mathbb{C}_t)$ for an inference step, which is again noted for future reference.

9.3.5 Axiom Download

This is the interesting case. Assume that a new blob subconfiguration $[B]\langle W\rangle$ is induced at time t as the result of a download of an axiom $C \in \mathit{Ax}^d(r)$. Then C must be one of the clauses inducing the subconfiguration, and we get that there are $\mathbb{C}_B \subseteq \mathbb{C}_{t-1}$ and $S \subseteq G \setminus B$ with $W = S \cap \mathit{lpp}(B)$ such that

$$\mathbb{C}_B \cup \{C\} \cup \mathbb{B}(S) \rhd \mathit{All}^+(B)\ . \tag{9.10}$$

Our intuition is that download of an axiom clause $C \in \mathit{Ax}^d(r)$ in the resolution derivation should correspond to an introduction of $[r]\langle \mathit{pred}(r)\rangle$ in the induced blob-pebbling. We want to prove that any other blob subconfiguration $[B]\langle W\rangle$ in $\mathbb{S}(\mathbb{C}_t)$ is derivable by the pebbling rules from $\mathbb{S}(\mathbb{C}_{t-1}) \cup [r]\langle \mathit{pred}(r)\rangle$. Also, we need to prove that the pebbling moves needed to go from $\mathbb{S}(\mathbb{C}_{t-1})$ to $\mathbb{S}(\mathbb{C}_t)$ do not increase the blob-pebbling cost by more than an additive constant compared to $\max\bigl\{\mathsf{cost}(\mathbb{S}(\mathbb{C}_{t-1})), \mathsf{cost}(\mathbb{S}(\mathbb{C}_t))\bigr\} = \mathsf{cost}(\mathbb{S}(\mathbb{C}_t))$, where the equality holds since no subconfigurations can disappear when we add clauses to the clause configuration.

We do the proof by a case analysis over r depending on where in the graph this vertex is located in relation to B. To simplify the proofs for the different cases, we first show a general technical lemma about pebble induction at axiom download.

Lemma 9.13. *Suppose that $\mathbb{C}_t = \mathbb{C}_{t-1} \cup C$ for $C \in \mathit{Ax}^d(r)$ and that $[B]\langle W\rangle$ is a new blob subconfiguration induced at time t as witnessed by* (9.10). *Then:*

1. *$r \notin S$.*
2. *$\mathit{pred}(r) \cap B = \emptyset$.*

3. *If $r \notin B$, then $\mathbb{C}_{t-1}$ induces $[B]\langle W \cup (\{r\} \cap \mathit{lpp}(B))\rangle$ if r is a source, and otherwise this subconfiguration can be derived from $\mathbb{S}(\mathbb{C}_{t-1})$ by inflation.*

4. *If r is a non-source vertex and $v \in \mathit{pred}(r)$ is such that $v \in \mathit{lpp}(B) \setminus S$, then we can derive $[B \cup v]\langle S \cap \mathit{lpp}(B \cup v)\rangle$ from $\mathbb{S}(\mathbb{C}_{t-1})$ by inflation.*

Proof. Suppose that $[B]\langle W\rangle \in \mathbb{S}(\mathbb{C}_t) \setminus \mathbb{S}(\mathbb{C}_{t-1})$. For part 1, noting that $\mathbb{B}(r) \vDash C$ for $C \in \mathit{Ax}^d(r)$ we see that $r \notin S$, as otherwise the implication (9.10) cannot be precise since C can be omitted.

If r is a source part 2 is trivial, so suppose $\mathit{pred}(r) = \{p, q\}$ and $C = \overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} r_l$. Then it follows from Lemma 9.11 that $\{p, q\} \cap B = \emptyset$.

For part 3, if r is a source, we have $C = \bigvee_{i=1}^{d} r_i$ and (9.10) becomes

$$\mathbb{C}_B \cup \mathbb{B}(S \cup r) \rhd \mathit{All}^{+}(B) \tag{9.11}$$

for $S \cup r \subseteq G \setminus B$, which shows that $\mathbb{C}_{t-1}$ induces

$$\begin{aligned}[B]\langle (S \cup r) \cap \mathit{lpp}(B)\rangle &= [B]\langle (S \cap \mathit{lpp}(B)) \cup (r \cap \mathit{lpp}(B))\rangle \\ &= [B]\langle (W \cup (r \cap \mathit{lpp}(B))\rangle \ .\end{aligned} \tag{9.12}$$

If r is a non-source we do not get a precise implication but still have

$$\mathbb{C}_B \cup \mathbb{B}(S \cup r) \vDash \mathit{All}^{+}(B) \tag{9.13}$$

and Lemma 9.12 yields that $[B]\langle (S \cup r) \cap \mathit{lpp}(B)\rangle = [B]\langle W \cup (r \cap \mathit{lpp}(B))\rangle$ is derivable by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$.

If $v \in \mathit{pred}(r)$ in part 4, the downloaded axiom can be written on the form $C = C' \vee \overline{v}_i$. Applying Lemma 8.17 on (9.10) we get

$$\mathbb{C}_B \cup \mathbb{B}(S) \vDash \mathit{All}^{+}(B) \vee v_i \subseteq \mathit{All}^{+}(B \cup v) \ . \tag{9.14}$$

By assumption, we have that $B \cup v$ is a chain and that $S \subseteq G \setminus (B \cup v)$, so Lemma 9.12 says that the subconfiguration $[B \cup v]\langle S \cap \mathit{lpp}(B \cup v)\rangle$ is derivable from $\mathbb{S}(\mathbb{C}_{t-1})$ by inflation. □

What we get from Lemma 9.13 is not in itself sufficient to derive the new blob subconfiguration $[B]\langle W\rangle$ in the blob-pebble game, but the lemma provides subconfigurations that will be used as building blocks in the derivations of $[B]\langle W\rangle$ below.

Now we are ready for the case analysis over the vertex r for the downloaded axiom clause $C \in \mathit{Ax}^d(r)$. Recall that the assumption is that there exists a blob subconfiguration $[B]\langle W\rangle \in \mathbb{S}(\mathbb{C}_t) \setminus \mathbb{S}(\mathbb{C}_{t-1})$ induced through (9.10) for $\mathbb{C}_B \subseteq \mathbb{C}_{t-1}$ and $S \subseteq G \setminus B$ with $W = S \cap \mathit{lpp}(B)$. Remember also that we want to explain all new subconfigurations in $\mathbb{S}(\mathbb{C}_t) \setminus \mathbb{S}(\mathbb{C}_{t-1})$ in terms of blob-pebbling moves from the set of subconfigurations $\mathbb{S}(\mathbb{C}_t) \cup \{[r]\langle \mathit{pred}(r)\rangle\}$. As illustrated in Figure 9.3, the cases for the vertex r are:

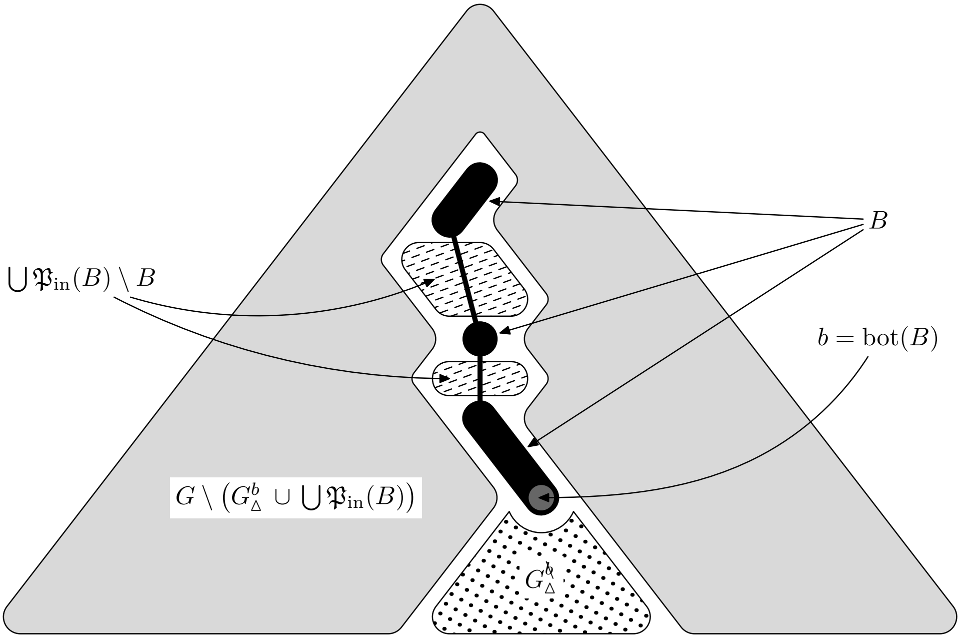

B
$\bigcup \mathfrak{P}_{\text{in}}(B) \setminus B$
$b = \text{bot}(B)$
$G \setminus \left(G^b_\Delta \cup \bigcup \mathfrak{P}_{\text{in}}(B)\right)$
G^b_Δ

Figure 9.3: Cases for vertex r w.r.t. blob B at download of axiom $C \in Ax^d(r)$.

1. $r \in G \setminus \left(G^b_\Delta \cup \bigcup \mathfrak{P}_{\text{in}}(B)\right)$ for $b = \text{bot}(B)$,
2. $r \in \bigcup \mathfrak{P}_{\text{in}}(B) \setminus B$,
3. $r \in B \setminus \{b\}$ for $b = \text{bot}(B)$,
4. $r = \text{bot}(B)$, and
5. $r \in G^b_\Delta$ for $b = \text{bot}(B)$.

Case 1: $r \in G \setminus \left(G^b_\Delta \cup \bigcup \mathfrak{P}_{\mathbf{in}}(B)\right)$ **for** $b = \text{bot}(B)$

If $r \in G \setminus \left(G^b_\Delta \cup \bigcup \mathfrak{P}_{\text{in}}(B)\right)$, this means that the vertex r is outside the set of vertices covered by source paths via B to top(B). In other words, $r \notin \mathit{lpp}(B) \cup B$ and part 3 of Lemma 9.13 yields that $\left[B\right]\left\langle W \cup (r \cap \mathit{lpp}(B))\right\rangle = [B]\langle W\rangle$ is derivable from $\mathbb{S}(\mathbb{C}_{t-1})$ by inflation. Note that we need no intermediate subconfigurations in this case.

Case 2: $r \in \bigcup \mathfrak{P}_{\mathbf{in}}(B) \setminus B$

This is the first more challenging case, and we do it in some detail to show how the reasoning goes. The proofs for the rest of the cases are analogous and will be presented in slightly more condensed form.

The condition $r \in \bigcup \mathfrak{P}_{\mathrm{in}}(B) \setminus B$ says that the vertex r is located on some path from $\mathrm{bot}(B)$ via B to $\mathrm{top}(B)$ strictly above the bottom vertex $b = \mathrm{bot}(B)$. In particular, this means that r cannot be a source vertex. Let $\mathit{pred}(r) = \{p, q\}$ and denote the downloaded axiom clause $C = \overline{p}_i \vee \overline{q}_j \vee \bigvee_{l=1}^{d} r_l$.

Part 3 of Lemma 9.13 says that we can derive the blob subconfiguration

$$[B]\langle W \cup (r \cap \mathit{lpp}(B))\rangle = [B]\langle W \cup r\rangle \tag{9.15}$$

by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$, where the equality holds since $r \in \bigcup \mathfrak{P}_{\mathrm{in}}(B) \setminus B \subseteq \mathit{lpp}(B)$ by Definition 9.6. Also, since r is on some path above b, at least one of the predecessors of r must be located on some path from b as well. That is, translating what was just said into our notation we have that the fact that $r \in \bigcup \mathfrak{P}_{\mathrm{in}}(B) \cap G_{\cancel{b}}^{\nabla}$ implies that either $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ or $q \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ or both. By symmetry, we get two cases: $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$, $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ and $\{p, q\} \subseteq \bigcup \mathfrak{P}_{\mathrm{in}}(B)$. Let us look at them in order.

I. $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$, $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(B)$: We make a subcase analysis depending on whether $p \in B \cup W$ or not. Recall from part 2 of Lemma 9.13 that $p \notin B$. The two remaining cases are $p \in W$ and $p \notin B \cup W$.

(a) $p \in W$: Let v be the uppermost vertex in B below p, or in formal notation

$$v = \mathrm{top}(G_{\Delta}^{p} \cap B) \ . \tag{9.16}$$

Such a vertex v must exist since $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B) \setminus B$. Since p is above v and is a predecessor of r, it lies on some path from v to r, i.e., $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(\{v, r\}) \setminus \{v, r\}$. For the sibling q we have $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(\{v, r\})$. This is so since $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ and for any path $P \in \mathfrak{P}_{\mathrm{in}}(\{v, r\})$ it holds that $P \subseteq \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ since there is nothing in between v and r in B, i.e., $\left(\bigcup \mathfrak{P}_{\mathrm{in}}(\{v, r\}) \setminus \{v, r\}\right) \cap B = \emptyset$. Also, $q \notin G_{\Delta}^{\cancel{p}} \supseteq G_{\Delta}^{\cancel{v}}$ because of the Sibling non-reachability property 9.4. Hence, it must hold that $q \notin \mathit{lpp}(\{v, r\})$.

We can use this information to make blob-pebbling moves resulting in $[B]\langle W\rangle$ as follows. First introduce $[r]\langle p, q\rangle$ and inflate this subconfiguration to

$$[v, r]\langle \{p, q\} \cap \mathit{lpp}(\{v, r\})\rangle = [v, r]\langle p\rangle \ . \tag{9.17}$$

Then derive the subconfiguration $[B]\langle W \cup r\rangle$ in (9.15) by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$. Finally, merge the two subconfigurations (9.15) and (9.17). The result of this merger move is $[B \cup v]\langle W \cup p\rangle = [B]\langle W\rangle$.

(b) $p \notin B \cup W$: Note that $p \in \mathfrak{P}_{\mathrm{in}}(B) \setminus B$ by assumption. Also, it must hold that $p \notin S$ since otherwise we would get the contradiction $p \in S \cap (\mathfrak{P}_{\mathrm{in}}(B) \setminus B) \subseteq S \cap \mathit{lpp}(B) = W$. Thus, $p \in \mathit{lpp}(B) \setminus S$ and part 4 of Lemma 9.13 yields that we can derive the blob subconfiguration

$$[B \cup p]\langle W_p\rangle \ \text{ for } \ W_p \subseteq W \tag{9.18}$$

by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$, where $W_p = S \cap \mathit{lpp}(B \cup p) \subseteq S \cap \mathit{lpp}(B) = W$ since $\mathit{lpp}(B \cup p) \subseteq \mathit{lpp}(B)$ if $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$. (This last claim is easily verified directly from Definition 9.6.)

With $v = \mathrm{top}(G^p_\Delta \cap B)$ as in (9.16), introduce $[r]\langle p,q\rangle$ and inflate to $[v,r]\langle p\rangle$ as in (9.17). Merging the subconfigurations (9.17) and (9.18) yields

$$[B \cup \{v,r\}]\langle W_p\rangle = [B \cup r]\langle W_p\rangle \tag{9.19}$$

and a second merger of the resulting subconfiguration (9.19) with the subconfiguration in (9.15) produces $[B]\langle W \cup W_p\rangle = [B]\langle W\rangle$.

This finishes the case $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$, $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(B)$.

II. $\{p,q\} \subseteq \bigcup \mathfrak{P}_{\mathrm{in}}(B)$: By part 2 of Lemma 9.13 $\{p,q\} \cap B = \emptyset$, so $\{p,q\} \subseteq \mathfrak{P}_{\mathrm{in}}(B) \setminus B$. By symmetry, we have the following subcases for p and q with respect to membership in B and W.

(a) $\{p,q\} \subseteq W$,

(b) $p \in W,\ q \notin W$,

(c) $\{p,q\} \cap (B \cup W) = \emptyset$.

We analyze these subcases one by one.

(a) $\{p,q\} \subseteq W$: This is easy. Just introduce $[r]\langle p,q\rangle$ and merge this subconfiguration with the subconfiguration (9.15) to get $[B]\langle W \cup \{p,q\}\rangle = [B]\langle W\rangle$.

(b) $p \in W,\ q \notin W$: In this case it must hold that $q \notin S$ since otherwise we would have $q \in S \cap (\mathfrak{P}_{\mathrm{in}}(B) \setminus B) \subseteq S \cap \mathit{lpp}(B) = W$ contradicting the assumption. Thus $q \in (\mathfrak{P}_{\mathrm{in}}(B) \setminus B) \setminus S \subseteq \mathit{lpp}(B) \setminus S$ and part 4 of Lemma 9.13 allows us to derive

$$[B \cup q]\langle W_q\rangle \ \text{ for } \ W_q \subseteq W \tag{9.20}$$

by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$. Here we have $W_q = S \cap \mathit{lpp}(B \cup q) \subseteq S \cap \mathit{lpp}(B) = W$ since $\mathit{lpp}(B \cup q) \subseteq \mathit{lpp}(B)$ when $q \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$.

Introduce $[r]\langle p,q\rangle$ and merge with the subconfiguration (9.20) to get

$$[B \cup r]\langle W_q \cup p\rangle \tag{9.21}$$

and then merge (9.21) with the subconfiguration $[B]\langle W \cup r\rangle$ from (9.15) to get $[B]\langle W \cup W_q \cup p\rangle = [B]\langle W\rangle$.

(c) $\{p, q\} \cap B \cup W = \emptyset$: Just as for the vertex q in case case IIb, here it holds for both p and q that $\{p, q\} \subseteq \mathit{lpp}(B) \setminus S$. Part 4 of Lemma 9.13 yields subconfigurations $[B \cup p]\langle W_p \rangle$ for $W_p \subseteq W$ as in (9.18) and $[B \cup q]\langle W_q \rangle$ for $W_q \subseteq W$ as in (9.20) derived by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$.

Introduce $[r]\langle p, q \rangle$ and merge with (9.18) on p to get

$$[B \cup r]\langle W_p \cup q \rangle \tag{9.22}$$

and then merge (9.22) with (9.20) on q resulting in

$$[B \cup r]\langle W_p \cup W_q \rangle \ . \tag{9.23}$$

Finally, merge (9.23) with (9.15) on r to get $[B]\langle W \cup W_p \cup W_q \rangle = [B]\langle W \rangle$.

This concludes the case $r \in \bigcup \mathfrak{P}_{\mathrm{in}}(B) \setminus B$. We can see that in all subcases, the new blob subconfiguration $[B]\langle W \rangle$ is derivable from $\mathbb{S}(\mathbb{C}_{t-1}) \cup [r]\langle \mathit{pred}(r) \rangle$ by inflation moves followed by mergers on some subset of $\{p, q, r\}$.

Let us analyze the cost of deriving $[B]\langle W \rangle$. We want to bound the cost of the intermediate subconfigurations that are used in the transition from $\mathbb{S}(\mathbb{C}_{t-1})$ to $\mathbb{S}(\mathbb{C}_t)$ but are not present in $\mathbb{S}(\mathbb{C}_t)$. We first note that for the subconfigurations $[B]\langle W \cup r \rangle$, $[B \cup p]\langle W_p \rangle$, $[B \cup q]\langle W_q \rangle$ and $[B \cup r]\langle W' \rangle$ for various $W' \subseteq W$, the chargeable vertices are all subsets of the chargeable vertices of the final subconfiguration $[B]\langle W \rangle$. This is so since $b = \mathrm{bot}(B)$ is the bottom vertex in all these black blobs, and all chargeable white vertices are contained in $W \cap G^{b}_{\triangle}$. The subconfigurations $[r]\langle p, q \rangle$ and $[v, r]\langle p \rangle$ for $v = \mathrm{top}(G^{p}_{\triangle} \cap B)$ can incur an extra cost, however, but this cost is clearly bounded by $|\{p, q, r, v\}| = 4$.

Case 3: $r \in B \setminus \{b\}$ **for** $b = \mathrm{bot}(B)$

First we note that in this case, we can no longer use part 3 of Lemma 9.13 to derive the blob subconfiguration $[B]\langle W \cup r \rangle$ of (9.15). The vertex r cannot be added to the support S since it is contained in B. Also, we note that r cannot be a source since it is above the bottom vertex b. As usual, let us write $\mathit{pred}(r) = \{p, q\}$.

Observe that just as in case 2 (Section 9.3.5) we must have either $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ or $q \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ or both. By symmetry we get the same two cases for membership of p and q in $\bigcup \mathfrak{P}_{\mathrm{in}}(B)$, namely $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$, $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ and $\{p, q\} \subseteq \bigcup \mathfrak{P}_{\mathrm{in}}(B)$.

I. $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(B)$, $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(B)$: As before, $p \notin B$ by part 2 of Lemma 9.13. We make a subcase analysis depending on whether $p \in W$ or $p \notin B \cup W$.

As in (9.16) we let $v = \mathrm{top}(G^{p}_{\triangle} \cap B)$ and note that $p \in \bigcup \mathfrak{P}_{\mathrm{in}}(\{v, r\}) \setminus \{v, r\}$. For q we have $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(\{v, r\})$ since $q \notin \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ but $\{v, r\} \subseteq \bigcup \mathfrak{P}_{\mathrm{in}}(B)$ and there is nothing in between v and r in B. Also, $q \notin G^{\bcancel{p}}_{\triangle} \supseteq G^{\bcancel{v}}_{\triangle}$ because of the Sibling non-reachability property 9.4. Hence, it holds that $q \notin \mathit{lpp}(\{v, r\})$.

(a) $p \in W$: Introduce $[r]\langle p, q\rangle$ and inflate to $[v, r]\langle \{p, q\} \cap \mathit{lpp}(\{v, r\})\rangle = [v, r]\langle p\rangle$ as in (9.17) and continue the inflation to $[B \cup \{v, r\}]\langle W \cup p\rangle = [B]\langle W\rangle$.

(b) $p \notin B \cup W$: Just as in case 2, $p \notin W$ implies $p \notin S$, so $p \in \mathit{lpp}(B) \setminus S$ and we can use part 4 of Lemma 9.13 to derive $[B \cup p]\langle W_p\rangle$ for $W_p \subseteq W$ as in (9.18). Introduce $[r]\langle p, q\rangle$, inflate to $[v, r]\langle p\rangle$ as in (9.17) and merge (9.17) and (9.18) on p resulting in $[B \cup \{v, r\}]\langle W_p\rangle = [B]\langle W_p\rangle$, which can be inflated to $[B]\langle W\rangle$.

II. $\{p, q\} \subseteq \bigcup \mathfrak{P}_{\mathrm{in}}(B)$: We have the same possibilities to consider for containment of p and q in $B \cup W$ as in case 2(II) on page 173.

(a) $\{p, q\} \subseteq W$: This is immediate. Introduce the subconfiguration $[r]\langle p, q\rangle$ and inflate to $[B \cup r]\langle W \cup \{p, q\}\rangle = [B]\langle W\rangle$.

(b) $p \in W$, $q \notin B \cup W$: Apply part 4 of Lemma 9.13 to derive $[B \cup q]\langle W_q\rangle$ for $W_q \subseteq W$ by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$. Then introduce $[r]\langle p, q\rangle$ and merge on q to get the subconfiguration $[B \cup r]\langle W_q \cup p\rangle = [B]\langle W_q \cup p\rangle$, which can be inflated further to $[B]\langle W_q \cup p \cup W\rangle = [B]\langle W\rangle$.

(c) $\{p, q\} \cap (B \cup W) = \emptyset$: In the same way as in case IIb, derive the subconfigurations $[B \cup p]\langle W_p\rangle$ and $[B \cup q]\langle W_q\rangle$ with $W_p \cup W_q \subseteq W$ from $\mathbb{S}(\mathbb{C}_{t-1})$ by inflation. Introduce $[r]\langle p, q\rangle$ and merge twice, first on p and then on q, to get $[B]\langle W_p \cup W_q\rangle$, which can be inflated to $[B]\langle W\rangle$.

This concludes the case $r \in B \setminus \{b\}$. We see that in all subcases the new blob subconfiguration $[B]\langle W\rangle$ is derivable from $\mathbb{S}(\mathbb{C}_{t-1}) \cup [r]\langle \mathit{pred}(r)\rangle$ by inflation moves followed by mergers on some subset of $\{p, q\}$, possibly followed by one more inflation move.

As in the previous case, the bottom vertex in all of the black blobs $[B \cup p]$, $[B \cup q]$ and $[B \cup r]$ is $b = \mathrm{bot}(B)$, and the corresponding chargeable white pebbles are subsets of those of W. The extra cost caused by the subconfigurations $[r]\langle p, q\rangle$ and $[v, r]\langle p\rangle$ is at most 4.

Case 4: $r = \mathrm{bot}(B)$

If r is a source, any $[B]\langle W\rangle$ with $r \in B$ can be derived by introducing $[r]\langle \mathit{pred}(r)\rangle = [r]\langle \emptyset\rangle$ and inflating. Suppose therefore that $r = \mathrm{bot}(B)$ is not a source and let $\mathit{pred}(r) = \{p, q\}$. Then it holds that $\{p, q\} \subseteq G^{\nwarrow}_{\triangle} \subseteq \mathit{lpp}(B)$, i.e., the vertex sets $B \cup p$ and $B \cup q$ are both chains.

By symmetry, we have three cases for p and q with respect to membership in W. (It is still true that $\{p, q\} \cap B = \emptyset$ by part 2 of Lemma 9.13.)

(a) $\{p, q\} \subseteq W$: Immediate. Introduce the subconfiguration $[r]\langle p, q\rangle$ and inflate it to $[B \cup r]\langle W \cup \{p, q\}\rangle = [B]\langle W\rangle$.

(b) $p \in W$, $q \notin W$: Enlist the help of our old friend Lemma 9.13, part 4, to derive $[B \cup q]\langle W_q \rangle$ for $W_q \subseteq W$ by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$ (where $W_q \subseteq W$ holds since $lpp(B \cup v) \subseteq lpp(B)$ if $v \in G^{\mathfrak{b}}_{\triangle}$). Introduce $[r]\langle p, q \rangle$ and merge with $[B \cup q]\langle W_q \rangle$ to get $[B \cup r]\langle W_q \cup p \rangle = [B]\langle W_q \cup p \rangle$. Then inflate $[B]\langle W_q \cup p \rangle$ to $[B]\langle W_q \cup p \cup W \rangle = [B]\langle W \rangle$.

(c) $\{p, q\} \cap W = \emptyset$: Following an established tradition, mimic case b and derive $[B \cup p]\langle W_p \rangle$ and $[B \cup q]\langle W_q \rangle$ with $W_p \cup W_q \subseteq W$ by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$. Introduce $[r]\langle p, q \rangle$, do two mergers to get $[B]\langle W_p \cup W_q \rangle$ and inflate to $[B]\langle W \rangle$.

This takes care of the case $r = b$. Again, in all subcases our new subconfiguration $[B]\langle W \rangle$ is derivable from $\mathbb{S}(\mathbb{C}_{t-1}) \cup [r]\langle pred(r) \rangle$ by inflation moves followed by mergers on some subset of $\{p, q\}$, possibly followed by one more inflation move.

This time the blobs $[B \cup p]$ and $[B \cup q]$ can cause an extra intermediate cost of 1 each for the bottom vertices p and q, and $[r]\langle p, q \rangle$ potentially adds an extra cost 1 for r, giving that the intermediate extra cost is bounded by 3.

Case 5: $r \in G^{\mathfrak{b}}_{\triangle}$ for $b = \mathrm{bot}(B)$

This final case is very similar to the previous case $r = \mathrm{bot}(B)$. Note first that $r \in G^{\mathfrak{b}}_{\triangle} \subseteq lpp(B)$. If r is a source, then $C = \bigvee_{i=1}^{d} r_i$ and we have

$$\mathbb{C}_B \cup \{C\} \cup \mathbb{B}(S) = \mathbb{C}_B \cup \mathbb{B}(S \cup r) \rhd All^{+}(B) \tag{9.24}$$

at time $t - 1$, which shows that $[B]\langle W \cup r \rangle \in \mathbb{S}(\mathbb{C}_{t-1})$. Hence, we can introduce $[r]\langle pred(r) \rangle = [r]\langle \emptyset \rangle$ and merge on r to get $[B]\langle W \rangle$.

As usual, the more interesting case is when r is a non-source with $pred(r) = \{p, q\}$. The case analysis is just as in case 4 (Section 9.3.5). However, note that now we can again use part 3 of Lemma 9.13 to derive $[B]\langle W \cup r \rangle$ from $\mathbb{S}(\mathbb{C}_{t-1})$ by inflation since it holds that $r \notin B$.

(a) $\{p, q\} \subseteq W$: Introducing $[r]\langle p, q \rangle$ and merging with $[B]\langle W \cup r \rangle$ yields $[B]\langle W \rangle$.

(b) $p \in W$, $q \notin W$: Appeal to part 4 of Lemma 9.13 to get $[B \cup q]\langle W_q \rangle$ for $W_q \subseteq W$ by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$. Introduce $[r]\langle p, q \rangle$ and merge to get $[B \cup r]\langle W_q \cup p \rangle$, and merge again with $[B]\langle W \cup r \rangle$ to get $[B]\langle W \rangle$.

(c) $\{p, q\} \cap W = \emptyset$: As in case b above for q, derive $[B \cup p]\langle W_p \rangle$ and $[B \cup q]\langle W_q \rangle$ with $W_p \cup W_q \subseteq W$ by inflation from $\mathbb{S}(\mathbb{C}_{t-1})$. Introduce $[r]\langle p, q \rangle$ and do two mergers to get $[B \cup r]\langle W_p \cup W_q \rangle$. Finally merge $[B \cup r]\langle W_p \cup W_q \rangle$ with $[B]\langle W \cup r \rangle$ to get $[B]\langle W \rangle$.

This takes care of the case $r = G^{\mathfrak{b}}_{\triangle}$. We note that in all subcases of this case, $[B]\langle W \rangle$ is derivable from $\mathbb{S}(\mathbb{C}_{t-1}) \cup [r]\langle pred(r) \rangle$ by inflation moves followed by mergers on some subset of $\{p, q, r\}$. Again, the extra intermediate pebbling cost is bounded by $|\{p, q, r\}| = 3$.

9.3.6 Wrapping up the Proof of Theorem 9.10

If $\pi = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ is a derivation of $\bigvee_{i=1}^{d} z_i$ from $*Peb_G^d$, it is easily verified from Definition 9.9 that $\mathbb{S}(\mathbb{C}_0) = \mathbb{S}(\emptyset) = \emptyset$ and $\mathbb{S}(\mathbb{C}_\tau) = \mathbb{S}(\{\bigvee_{i=1}^{d} z_i\}) = \{[z]\langle\emptyset\rangle\}$.

In Sections 9.3.3, 9.3.4, and 9.3.5, we have shown how to do the intermediate blob-pebbling moves to get from $\mathbb{S}(\mathbb{C}_{t-1})$ to $\mathbb{S}(\mathbb{C}_t)$ in the case of erasure, inference and axiom download, respectively. For erasure and inference, the blob-pebbling cost changes monotonically during the transition $\mathbb{S}(\mathbb{C}_{t-1}) \rightsquigarrow \mathbb{S}(\mathbb{C}_t)$. In the case of axiom download, there can be an extra cost of 4 incurred for deriving each $[B]\langle W\rangle \in \mathbb{S}(\mathbb{C}_t) \setminus \mathbb{S}(\mathbb{C}_{t-1})$. We have no a priori upper bound on $\left|\mathbb{S}(\mathbb{C}_t) \setminus \mathbb{S}(\mathbb{C}_{t-1})\right|$, but if we just derive the new subconfigurations one by one and erase all intermediate subconfigurations in between these derivations, we will keep the total extra cost below 4.

This shows that the complete blob-pebbling $\mathcal{P}_\pi$ of G associated to a resolution derivation $\pi : *Peb_G^d \vdash \bigvee_{i=1}^{d} z_i$ by the construction in this section has blob-pebbling cost bounded from above by $\mathit{cost}(\mathcal{P}_\pi) \leq \max_{\mathbb{C}\in\pi}\{\mathit{cost}(\mathbb{S}(\mathbb{C}))\} + 4$. Theorem 9.10 is thereby proven.

9.4 Induced Blob Configurations and Clause Set Size

In this section we prove that if a set of clauses $\mathbb{C}$ induces a blob-pebbling configuration $\mathbb{S}(\mathbb{C})$ according to Definition 9.9, then the cost of $\mathbb{S}(\mathbb{C})$ as specified in Definition 9.8 is at most $|\mathbb{C}|$ provided that $d \geq 2$. That is, the cost of an induced blob-pebbling configuration provides a lower bound on the size of the set of clauses inducing it.

To simplify the proofs, we will reuse the notation $Vars^d(U)$, $V(\mathbb{C})$, and $\mathbb{C}[\![U]\!]$ from Definition 8.28 on page 116. Let us also recall some technical results about CNF formulas that will come in handy in the proof of Theorem 9.16. Intuitively, we will use Lemma 8.24 together with Lemma 8.26 to argue that if a clause set $\mathbb{C}$ induces a lot of subconfigurations, then there must be a lot of variable occurrences in $\mathbb{C}$ for variables corresponding to these vertices. We will also need the fact that a minimally unsatisfiable CNF formula must have more clauses than variables, which we restate here for reference.

Theorem 8.27 (restated). *Suppose that F implies D minimally. For any subset of variables V of F, let $F_V = \{C \in F \mid Vars(C) \cap V \neq \emptyset\}$ denote the set of clauses containing variables from V. Then if $V \subseteq Vars(F) \setminus Vars(D)$, it holds that $|F_V| > |V|$.*

Given that Lemmas 8.24 and 8.26 tell us that many induced subconfigurations implies the presence of many variables in $\mathbb{C}$, we will use Theorem 8.27 to demonstrate that a lot of different variable occurrences will have to translate into a lot of different clauses, provided that the pebbling degree d is at least 2. Before we prove this formally, let us try to provide some intuition for why it should be true by studying two special cases. Recall the notation $\mathbb{B}(V) = \bigl\{\bigvee_{i\in[d]} v_i \bigm| v \in V\bigr\}$ and $\mathit{All}^{+}(V) = \bigvee_{v\in V} \bigvee_{i\in[d]} v_i$ from Definition 8.12.

Example 9.14. Suppose that $\mathbb{C}$ is a clause set derived from $^{*}\!\mathit{Peb}^{d}_{G}$ that induces N independent black blobs $B_1, \ldots, B_N$ that are pairwise disjoint, i.e., $B_i \cap B_j = \emptyset$ if $i \neq j$. Then the implications

$$\mathbb{C} \vDash \mathit{All}^{+}(B_i) \tag{9.25}$$

hold for $i = 1, \ldots, N$. Remember that since $^{*}\!\mathit{Peb}^{d}_{G}$ is non-contradictory, so is $\mathbb{C}$.

It is clear that a non-contradictory clause set $\mathbb{C}$ satisfying (9.25) for $i = 1, \ldots, N$ is quite simply the set

$$\mathbb{C} = \bigl\{\mathit{All}^{+}(B_i) \bigm| i = 1, \ldots N\bigr\} \tag{9.26}$$

consisting precisely of the clauses implied. Also, it seems plausible that this is the best one can do. Informally, if there would be strictly fewer clauses than N, some clause would have to mix variables from different blobs B_i and B_j. But then Lemma 8.26 says that there will be extra clauses needed to "neutralize" the literals from B_j in the implication $\mathbb{C} \vDash \mathit{All}^{+}(B_i)$ and vice versa, so that the total number of clauses would have to be strictly greater than N.

As it turns out, the proof that $|\mathbb{C}| \geq N$ when $\mathbb{C}$ induces N pairwise disjoint and independent black blobs is very easy. Suppose on the contrary that (9.25) holds for $i = 1, \ldots, N$ but that $|\mathbb{C}| < N$. Let α be a satisfying assignment for $\mathbb{C}$. Choose $\alpha' \subseteq \alpha$ to be any minimal partial truth value assignment fixing $\mathbb{C}$ to true. Then for the size of the domain of α' we have $|\mathrm{Dom}(\alpha')| < N$, since at most one distinct literal is needed for every clause $C \in \mathbb{C}$ to fix it to true. This means that there is some B_i such that α' does not set any variables in $\mathit{Vars}^{d}(B_i)$. Consequently α' can be extended to an assignment α'' setting $\mathbb{C}$ to true but $\mathit{All}^{+}(B_i)$ to false, which is a contradiction. With some more work, and using Theorem 8.27, one can show that $|\mathbb{C}| > N$ if variables from distinct blobs are mixed.

Note that the above argument works for any pebbling degree including $d = 1$. Intuitively, this means that one can charge for black blobs even in the case of first degree pebbling formulas.

Example 9.15. Suppose that the clause set $\mathbb{C}$ induces an blob subconfiguration $[B]\langle W\rangle$ with $W \neq \emptyset$, and let us assume for simplicity that $\mathbb{C}$ is minimal and $W = S$ so that the implication

$$\mathbb{C} \cup \mathbb{B}(W) \vDash \mathit{All}^{+}(B) \tag{9.27}$$

holds and is minimal. We claim that $|\mathbb{C}| \geq |W| + 1$ provided that $d > 1$.

Since by definition $B \cap W = \emptyset$ we have $\mathit{Vars}(\mathit{All}^{+}(B)) \cap \mathit{Vars}(\mathbb{B}(W)) = \emptyset$, and Theorem 8.27 yields that $|\mathbb{C} \cup \mathbb{B}(W)| \geq |\mathbb{C}[\![W]\!] \cup \mathbb{B}(W)| > |\mathit{Vars}(\mathbb{B}(W))|$. This is not quite what we want—we have a lower bound on $|\mathbb{C} \cup \mathbb{B}(W)|$, but what we need is a bound on $|\mathbb{C}|$. But if we observe that $|\mathit{Vars}(\mathbb{B}(W))| = d|W|$ while $|\mathbb{B}(W)| = |W|$, we get the claimed inequality

$$|\mathbb{C}| \geq |\mathit{Vars}(\mathbb{B}(W))| - |\mathbb{B}(W)| + 1 = (d-1)|W| + 1 \geq |W| + 1 \ . \tag{9.28}$$

We remark that this time we had to use that $d > 1$ in order to get a lower bound on the clause set size. And indeed, it is not hard to see that a single clause on the form $C = v_1 \vee \bigvee_{w \in W} \overline{w}_1$ can induce an arbitrary number of white pebbles if $d = 1$. Intuitively, white pebbles can be had for free in first degree pebbling formulas.

In general, matters are more complicated than in Examples 9.14 and 9.15. If $[B_1]\langle W_1 \rangle$ and $[B_2]\langle W_2 \rangle$ are two induced blob subconfigurations, the black blobs B_1 and B_2 need not be disjoint, the supporting white pebbles W_1 and W_2 might also intersect, and the black blob B_1 can intersect the supporting white pebbles W_2 of the other blob. Nevertheless, if we choose with some care which vertices to charge for, the intuition provided by our examples can still be used to prove the following theorem (which is the parallel of Theorem 8.29).

Theorem 9.16. *Suppose that G is a blob-pebblable DAG and let $\mathbb{C}$ be a set of clauses derived from the pebbling formula *Peb^d_G for $d \geq 2$. Then $|\mathbb{C}| \geq \mathsf{cost}(\mathbb{S}(\mathbb{C}))$.*

Proof. Let $\mathbb{S}(\mathbb{C}) = \{[B_i]\langle W_i \rangle \,|\, i \in [m]\}$ be set set of induced blob subconfigurations. By Definition 9.8, we have $\mathsf{cost}(\mathbb{S}(\mathbb{C})) = \left|\mathcal{B} \cup \mathcal{W}^{\Delta}\right|$ where

$$\mathcal{B} = \left\{\mathrm{bot}(B_i) \,\middle|\, [B_i]\langle W_i \rangle \in \mathbb{S}(\mathbb{C})\right\} \tag{9.29}$$

and

$$\mathcal{W}^{\Delta} = \bigcup_{[B_i]\langle W_i \rangle \in \mathbb{S}(\mathbb{C})} \left(W_i \cap G_{\Delta}^{\mathrm{bot}(B_i)}\right) \; . \tag{9.30}$$

We need to prove that $|\mathbb{C}| \geq \left|\mathcal{B} \cup \mathcal{W}^{\Delta}\right|$.

We first show that all vertices in $\mathcal{B} \cup \mathcal{W}^{\Delta}$ are represented in some clause in $\mathbb{C}$. By Definition 9.9, for each $[B_i]\langle W_i \rangle \in \mathbb{S}(\mathbb{C})$ there is a clause set $\mathbb{C}_i \subseteq \mathbb{C}$ and a vertex set $S_i \subseteq G \setminus B_i$ with $W_i = S_i \cap \mathit{lpp}(B_i) \subseteq S_i$ such that

$$\mathbb{C}_i \cup \mathbb{B}(S_i) \vDash \mathit{All}^+(B_i) \tag{9.31}$$

and such that this implication does not hold for any strict subset of $\mathbb{C}_i$, S_i or B_i. Fix (arbitrarily) such $\mathbb{C}_i$ and S_i for every $[B_i]\langle W_i \rangle \in \mathbb{S}(\mathbb{C})$ for the rest of this proof.

For the induced black blobs B_i we claim that $B_i \subseteq V(\mathbb{C}_i)$, which certainly implies $\mathrm{bot}(B_i) \in V(\mathbb{C})$. To establish this claim, note that for any $v \in B_i$ we can apply Lemma 8.24 with $D_1 = \bigvee_{j=1}^{d} v_j$ and $D_2 = \mathit{All}^+(B_i \setminus \{v\})$ on the implication (9.31), which yields that the vertex v must be represented in $\mathbb{C}_i \cup \mathbb{B}(W_i)$ by some positive literal v_j. Since $B_i \cap S_i = \emptyset$, we have $\mathit{Vars}(\mathbb{B}(S_i)) \cap \mathit{Vars}(\mathit{All}^+(B_i)) = \emptyset$ and thus $v_j \in \mathit{Lit}(\mathbb{C}_i)$.

Also, we claim that $S_i \subseteq V(\mathbb{C}_i)$. To see this, note that since $B_i \cap S_i = \emptyset$ and the implication (9.31) is minimal, it follows from Lemma 8.26 that for every $w \in S_i$, all literals $\overline{w}_j$, $j \in [d]$, must be present in $\mathbb{C}_i$. Thus, in particular, it holds that $W_i \cap G_{\Delta}^{\mathrm{bot}(B_i)} \subseteq V(\mathbb{C}_i)$.

We now prove by induction over subsets $R \subseteq \mathcal{B} \cup \mathcal{W}^{\Delta}$ that $|\mathbb{C}[\![R]\!]| \geq |R|$. The theorem clearly follows from this since $|\mathbb{C}| \geq |\mathbb{C}[\![R]\!]|$. (The reader can think of R as the set of vertices *representing* the subconfigurations $[B_i]\langle W_i\rangle \in \mathbb{S}(\mathbb{C})$ in the clause set $\mathbb{C}$.)

The base case $|R| = 1$ is immediate, since we just demonstrated that all vertices $r \in R$ are represented in $\mathbb{C}$.

For the induction step, suppose that $|\mathbb{C}[\![R']\!]| \geq |R'|$ for all $R' \subsetneqq R$. Pick a "topmost" vertex $r \in R$, i.e., such that $G^{\nabla}_{\not{r}} \cap R = \emptyset$. We associate a blob subconfiguration $[B_i]\langle W_i\rangle \in \mathbb{S}(\mathbb{C})$ with r as follows. If $r = \mathrm{bot}(B_i)$ for some $[B_i]\langle W_i\rangle$, fix $[B_i]\langle W_i\rangle$ arbitrarily to such a subconfiguration. Otherwise, there must exist some $[B_i]\langle W_i\rangle$ such that $r \in W_i \cap G_{\vartriangle}^{\mathrm{bot}(B_i)}$, so fix any such subconfiguration. We note that it holds that

$$R \cap G^{\nabla}_{\mathrm{bot}(B_i)} \subseteq \{r\} \tag{9.32}$$

for $[B_i]\langle W_i\rangle$ chosen in this way.

Consider the clause set $\mathbb{C}_i \subseteq \mathbb{C}$ and vertex set $S_i \supseteq W_i$ from (9.31) associated with $[B_i]\langle W_i\rangle$ above. Clearly, by construction $r \in V(\mathbb{C}_i)$ is one of the vertices of R mentioned by $\mathbb{C}_i$. We claim that the total number of vertices in R mentioned by $\mathbb{C}_i$ is upper-bounded by the number of clauses in $\mathbb{C}_i$ mentioning these vertices, i.e., that

$$\left|\mathbb{C}_i[\![R]\!]\right| \geq \left|R \cap V(\mathbb{C}_i)\right| \ . \tag{9.33}$$

Let us first see that this claim is sufficient to prove the theorem. To this end, let

$$R[i] = R \cap V(\mathbb{C}_i) \tag{9.34}$$

denote the set of all vertices in R mentioned by $\mathbb{C}_i$ and assume that $|\mathbb{C}_i[\![R]\!]| = |\mathbb{C}_i[\![R[i]]\!]| \geq |R[i]|$. Observe that $\mathbb{C}_i[\![R[i]]\!] \subseteq \mathbb{C}[\![R]\!]$, since $\mathbb{C}_i \subseteq \mathbb{C}$ and $R[i] \subseteq R$. Or in words: the set of clauses in $\mathbb{C}_i$ mentioning vertices in $R[i]$ is certainly a subset of all clauses in $\mathbb{C}$ mentioning any vertex in R. Also, by construction $\mathbb{C}_i$ does not mention any vertices in $R \setminus R[i]$ since $R[i] = R \cap V(\mathbb{C}_i)$. That is,

$$\mathbb{C}[\![R \setminus R[i]]\!] \subseteq \mathbb{C}[\![R]\!] \setminus \mathbb{C}_i \tag{9.35}$$

in our notation. Combining the (yet unproven) claim (9.33) for $\mathbb{C}_i[\![R]\!] = \mathbb{C}_i[\![R[i]]\!]$ asserting that $\left|\mathbb{C}_i[\![R[i]]\!]\right| \geq |R[i]|$ with the induction hypothesis for $R \setminus R[i] \subseteq R \setminus \{r\} \subsetneqq R$, we get

$$\begin{aligned}
\left|\mathbb{C}[\![R]\!]\right| &= \left|\mathbb{C}_i[\![R]\!] \,\dot\cup\, (\mathbb{C} \setminus \mathbb{C}_i)[\![R]\!]\right| \\
&\geq \left|\mathbb{C}_i[\![R \cap V(\mathbb{C}_i)]\!] \,\dot\cup\, \mathbb{C}[\![R \setminus V(\mathbb{C}_i)]\!]\right| \\
&= \left|\mathbb{C}_i[\![R[i]]\!]\right| + \left|\mathbb{C}[\![R \setminus R[i]]\!]\right| \\
&\geq |R[i]| + |R \setminus R[i]| \\
&= |R|
\end{aligned} \tag{9.36}$$

and the theorem follows by induction.

It remains to verify the claim (9.33) that $|\mathbb{C}_i[\![R[i]]\!]| \geq |R[i]|$ for $R[i] = R \cap V(\mathbb{C}_i) \neq \emptyset$. To do so, recall first that $r \in R[i]$. Thus, $R[i] \neq \emptyset$ and if $R[i] = \{r\}$ we trivially have $|\mathbb{C}_i[\![R[i]]\!]| \geq 1 = |R[i]|$. Suppose therefore that $R[i] \supsetneqq \{r\}$.

We want to apply Theorem 8.27 on the formula $F = \mathbb{C}_i \cup \mathbb{B}(S_i)$ on the left-hand side of the minimal implication (9.31). Let $R' = R[i] \setminus \{r\}$, write $R' = R_1 \mathbin{\dot\cup} R_2$ for $R_1 = R' \cap S_i$ and $R_2 = R' \setminus R_1$, and consider the subformula

$$\begin{aligned} F_{R'} &= \bigl\{C \in \bigl(\mathbb{C}_i \cup \mathbb{B}(S_i)\bigr) \bigm| V(C) \cap R' \neq \emptyset\bigr\} \\ &= \mathbb{C}_i[\![R']\!] \cup \mathbb{B}(R_1) \end{aligned} \tag{9.37}$$

of $F = \mathbb{C}_i \cup \mathbb{B}(S_i)$. A key observation for the concluding part of the argument is that by (9.32) we have $\mathit{Vars}^d(R') \cap \mathit{Vars}(\mathit{All}^+(B_i)) = \emptyset$.

For each $w \in R_1$, the clauses in $\mathbb{B}(R_1)$ contain d literals $w_1, \ldots, w_d$ and these literals must all occur negated in $\mathbb{C}_i$ by Lemma 8.26. For each $u \in R_2$, the clauses in $\mathbb{C}_i[\![R']\!]$ contain at least one variable u_i. Appealing to Theorem 8.27 with the subset of variables $\mathit{Vars}^d(R') \cap \mathit{Vars}(\mathbb{C}_i) \subseteq \mathit{Vars}(F) \setminus \mathit{Vars}(\mathit{All}^+(B_i))$, we get

$$\begin{aligned} \bigl|F_{R'}\bigr| &= \bigl|\mathbb{C}_i[\![R']\!] \cup \mathbb{B}(R_1)\bigr| \\ &\geq \bigl|\,\mathit{Vars}^d(R') \cap \mathit{Vars}(\mathbb{C}_i)\bigr| + 1 \\ &\geq d\bigl|R_1\bigr| + \bigl|R_2\bigr| + 1 \ , \end{aligned} \tag{9.38}$$

and rewriting this as

$$\begin{aligned} \bigl|\mathbb{C}_i[\![R[i]]\!]\bigr| &\geq \bigl|\mathbb{C}_i[\![R']\!]\bigr| \\ &= \bigl|F_{R'}\bigr| - \bigl|\mathbb{B}(R_1)\bigr| \\ &\geq (d-1)\bigl|R_1\bigr| + \bigl|R_2\bigr| + 1 \\ &\geq \bigl|R[i]\bigr| \end{aligned} \tag{9.39}$$

establishes the claim. □

We have two concluding remarks. Firstly, we note that the place where the condition $d \geq 2$ is needed is the very final step (9.39). This is where an attempted lower bound proof for first degree pebbling formulas $*\mathit{Peb}^1_G$ would fail for the reason that the presence of many white pebbles in $\mathbb{S}(\mathbb{C})$ says absolutely nothing about the size of the clause set $\mathbb{C}$ inducing these pebbles. Secondly, another crucial step in the proof is that we can choose our representative vertices $r \in R$ so that (9.32) holds. It is thanks to this fact that the inequalities in (9.38) go through. The way we make sure that (9.32) holds is to charge only for (distinct) bottom vertices in the black blobs, and only for supporting white pebbles below these bottom vertices.

9.5 A Tight Bound for Blob-Pebbling the Pyramid

Having come this far in our proof construction, we know that resolution derivations induce blob-pebblings. We also know that blob-pebbling cost gives a lower bound on clause set size and hence on the space of the derivation. The final component needed to make the proof of Theorem 2.3 complete is to show lower bounds on the blob-pebbling price $\mathsf{Blob\text{-}Peb}(G_n)$ for some nice family of blob-pebblable directed acyclic graphs $\{G_n\}_{n=1}^{\infty}$.

Perhaps the first idea that comes to mind is to try to establish lower bounds on blob-pebbling price by reducing this problem to the problem of proving lower bounds for the standard black-white pebble game of Definition 5.1. This is what we did in Chapter 8 for the restricted case of trees. There, for the labelled pebblings $\mathcal{L}_\pi$ that one gets from resolution derivations $\pi : {}^*Peb_T^d \vdash \bigvee_{i=1}^d z_i$ in the labelled pebble game, we presented an explicit procedure for transforming any $\mathcal{L}_\pi$ into a complete black-white pebbling $\mathcal{P}$ of T in asymptotically the same cost. The lower bound on pebbling price in the labelled pebble game then followed by using the known lower bound for black-white pebbling of trees in Theorem 5.2.

Unfortunately, the blob-pebble game seems more difficult than the labelled pebble game to analyze in terms of standard black-white pebbling. The problem is the inflation rule (in combination with the cost function). It is not hard to show that without inflation, the blob-pebble game is essentially just a disguised form of black-white pebbling (this follows from Lemma 8.30). Thus, if we could convert any blob-pebbling into an equivalent pebbling not using inflation moves without increasing the cost by more than, say, some constant factor, we would be done. But in contrast to the case for the labelled pebble game in Chapter 8 played on binary trees, we are not able to transform blob-pebblings into black-white pebblings in a cost-preserving way.

Instead, what we do is to prove lower bounds directly for the blob-pebble game. This is not immediately clear how to do, since the lower bound proofs for black-white pebbling price in, for instance, [34, 42, 49, 55] all break down for the more general blob-pebble game. We are only able to obtain lower bounds for the limited class of blob-pebblable DAGs (Definition 9.5) that are also layered (Definition 6.2). We show that for all such DAGs G_h of height h that are spreading in the sense of Definition 6.44, it holds that $\mathsf{Blob\text{-}Peb}(G_h) = \Theta(h)$. In particular, this bound holds for pyramids Π_h since they are spreading by Theorem 6.45.

The constant factor that we get in our lower bound is moderately small and explicit. In fact, we believe that it should hold that $\mathsf{Blob\text{-}Peb}(G_h) \geq h/2 + \mathrm{O}(1)$ for layered spreading graphs G_h of height h, just as in the standard black-white pebble game. As we have not made any real attempt to get optimal constants, the factor in our lower bound can be improved with a minor effort, but additional ideas seems to be needed to push the constant all the way up to $\frac{1}{2}$.

9.5.1 Definitions and Notation for Blob-Pebbling Price Lower Bound

Recall that a vertex set U hides a black pebble on b if it blocks all source paths visiting v. For a blob B, which is a chain by Definition 9.6, it appears natural to extend this definition by requiring that U should block all paths going through all of B. We recall the terminology and notation from Definition 6.5 that a black blob B and a path P *agree* with each other, or that P is a path *via* B, if $B \subseteq P$, and that $\mathfrak{P}_{\mathrm{via}}(B)$ denotes the set of all source paths agreeing with B.

Definition 9.17 (Blocked black blob). A vertex set U *blocks* a blob B if U blocks all paths $P \in \mathfrak{P}_{\mathrm{via}}(B)$.

A terminological aside: Recalling the discussion in Section 6.5 it seems natural to say that U *blocks* a black blob B rather than hides it, since standing at the sources we might "see" the beginning of B, but if we try to walk any path via B we will fail before reaching the top of B since U blocks the path. This distinction between hiding and blocking turns out to be a very important one in our lower bound proof for blob-pebbling price. Of course, if B is an atomic black pebble, i.e., $|B| = 1$, the hiding and blocking relations coincide.

Let us next define what it means to block a blob-pebbling configuration.

Definition 9.18 (Unblocked paths). For $[B]\langle W\rangle$ an blob subconfiguration, the set of *unblocked paths* for $[B]\langle W\rangle$ is

$$\mathrm{unblocked}([B]\langle W\rangle) = \{P \in \mathfrak{P}_{\mathrm{via}}(B) \mid P \cap W = \emptyset\}$$

and we say that U blocks $[B]\langle W\rangle$ if U blocks all paths in $\mathrm{unblocked}([B]\langle W\rangle)$. We say that U blocks the blob-pebbling configuration $\mathbb{S}$ if U blocks all $[B]\langle W\rangle \in \mathbb{S}$. If so, we say that U is a *blocker* of $[B]\langle W\rangle$ or $\mathbb{S}$, respectively, or a *blocking set* for $[B]\langle W\rangle$ or $\mathbb{S}$.

Comparing to Section 6.5, note that when blocking a path $P \in \mathfrak{P}_{\mathrm{via}}(B)$, U can only use the white pebbles W that are associated with B in $[B]\langle W\rangle$. Although there might be white pebbles from other subconfigurations $[B']\langle W'\rangle \neq [B]\langle W\rangle$ that would be really helpful, U cannot enlist the help of the white pebbles in W' when blocking B. The reason for defining the blocking relation in this way is that these white pebbles can suddenly disappear due to pebbling moves performed on such subconfigurations $[B']\langle W'\rangle$.

Reusing the definition of measure in Definition 6.18 on page 67, we generalize the concept of *potential* to blob-pebbling configurations as follows.

Definition 9.19 (Blob-pebbling potential). The *potential* of a blob-pebbling configuration $\mathbb{S}$ is

$$\mathrm{pot}(\mathbb{S}) = \min\{m(U) : U \text{ blocks } \mathbb{S}\} \ .$$

If U is such that U blocks $\mathbb{S}$ and U has minimal measure $m(U)$ among all blocking sets for $\mathbb{S}$, we say that U is a *minimum-measure* blocking set for $\mathbb{S}$.

To compare blob-pebbling potential with the black-white pebbling potential in Definition 6.19, consider the following examples with vertex labels as in our running example pyramid in Figures 6.2 and 6.4–6.6.

Example 9.20. Consider the blob-pebbling configuration $\mathbb{S} = \bigl\{[z]\langle y_1\rangle, [z]\langle y_2\rangle\bigr\}$. Then the minimum-measure blocker for $\mathbb{S}$ is $U = \{z\}$, but the standard black-white pebble configuration $\mathbb{P} = (B, W) = (\{z\}, \{y_1, y_2\})$ with pebbles on the same vertices has $U = \emptyset$ as minimum-measure hiding set.

Example 9.21. For the blob-pebbling configuration $\mathbb{S} = \bigl\{[z]\langle\emptyset\rangle, [y_1]\langle x_1, x_2\rangle\bigr\}$, the minimum-measure blocker is again $U = \{z\}$. In comparison, for the standard black-white pebble configuration $\mathbb{P} = (B, W) = (\{z, y_1\}, \{x_1, x_2\})$ we have the minimum-measure hiding set $U = \{x_3\}$.

Remark 9.22. Perhaps it is also worth pointing out that Definition 9.19 is indeed a strict generalization of Definition 6.19. Given a black-white pebble configuration $\mathbb{P} = (B, W)$ we can construct an equivalent blob-pebbling configuration $\mathbb{S}(\mathbb{P})$ with respect to potential by setting

$$\mathbb{S}(\mathbb{P}) = \bigl\{ [b]\bigl\langle W \cap G^b_{\triangle}\bigr\rangle \big| b \in B \bigr\} \tag{9.40}$$

but as the examples above show going in the other direction is not possible.

Since we have accumulated a number of different minimality criteria for blocking sets, let us pause to clarify the terminology:

- The vertex set U is a *subset-minimal*, or just *minimal*, blocker for the blob-pebbling configuration $\mathbb{S}$ if no strict subset $U' \subsetneqq U$ is a blocking set for $\mathbb{S}$.
- U is a *minimum-measure* blocking set for $\mathbb{S}$ if it has minimal measure among all blocking sets for $\mathbb{S}$ (and thus yields the potential of $\mathbb{S}$).
- U is a *minimum-size* blocking set for $\mathbb{S}$ if it has minimal size among all blocking sets for $\mathbb{S}$.

Note that we can assume without loss of generality that minimum-measure and minimum-size blockers are both subset-minimal, since throwing away superfluous vertices can only decrease the measure and size, respectively. However, minimum-measure blockers need not have minimal size and vice versa. For a simple example of this, consider (with vertex labels as in Figures 6.2 and 6.4–6.6) the blob-pebbling configuration $\mathbb{S} = \bigl\{[z]\langle w_3, w_4\rangle\bigr\}$ and the two blocking sets $U_1 = \{z\}$ and $U_2 = \{w_1, w_2\}$.

9.5.2 A Lower Bound Assuming a Generalized LHC Property

For the blob-pebble game, a useful generalization of Property 6.21 on page 68 turns out to be the following.

Property 9.23 (Generalized limited hiding-cardinality property). We say that a blob-pebbling configuration $\mathbb{S}$ on a layered blob-pebblable DAG G has the *Generalized limited hiding-cardinality property with parameter C_K* if there is a vertex set U such that

1. U blocks $\mathbb{S}$,
2. $\mathrm{pot}(\mathbb{S}) = m(U)$, i.e., U is a minimum-measure blocker of $\mathbb{S}$,
3. $|U| \leq C_K \cdot \mathit{cost}(\mathbb{S})$.

For brevity, in what follows we will just refer to the *Generalized LHC property.*

We say that the graph G has the Generalized LHC property with parameter C_K if all blob-pebbling configurations $\mathbb{S}$ on G have the Generalized LHC property with parameter C_K.

When the parameter C_K is clear from context, we will just write that $\mathbb{S}$ or G has the Generalized LHC property.

For all layered blob-pebblable DAGs G_h of height h that have the Generalized LHC property and are spreading, it holds that $\textsf{\textit{Blob-Peb}}(G_h) = \Theta(h)$. The proof of this fact is very much in the spirit of the proofs of Lemma 6.22 and Theorem 6.24, although the details are slightly more complicated.

Theorem 9.24 (Analogue of Theorem 6.24). *Suppose that G_h is a layered blob-pebblable DAG of height h possessing the Generalized LHC property 9.23 with some parameter C_K. Then for any unconditional blob-pebbling $\mathcal{P} = \bigl\{\mathbb{S}_0 = \emptyset, \mathbb{S}_1, \ldots, \mathbb{S}_\tau\bigr\}$ of G_h it holds that*

$$\operatorname{pot}(\mathbb{S}_t) \leq (2C_K + 1) \cdot \max_{s \leq t}\{\textsf{\textit{cost}}(\mathbb{S}_s)\} \ . \tag{9.41}$$

In particular, for any family of layered blob-pebblable DAGs G_h that are also spreading in the sense of Definition 6.44, we have $\textsf{\textit{Blob-Peb}}(G_h) = \Theta(h)$.

We make two separate observations before presenting the proof.

Observation 9.25. *For any layered DAG G_h of height h, $\textsf{\textit{Blob-Peb}}(G_h) = \mathrm{O}(h)$.*

Proof. Any layered DAG G_h can be black-pebbled with $h + \mathrm{O}(1)$ pebbles by Theorem 6.9 on page 61, and it is easy to see that a blob-pebbling can mimic a black pebbling in the same cost. □

Observation 9.26. *If G_h is a layered blob-pebblable DAG of height h that is spreading in the sense of Definition 6.44, then $\operatorname{pot}_{G_h}([z]\langle\emptyset\rangle) = h + 2$.*

Proof. This is fairly similar to the corresponding case for pyramids in Lemma 6.23. Note, though, that in contrast to Lemma 6.23, here we cannot get the statement from the Generalized LHC property, but instead have to prove it directly.

Since $[z]$ is an atomic blob, the blocking and hiding relations coincide. The set $U = \{z\}$ hides itself and has measure $h + 2$. We show that any other blocking set must have strictly larger measure.

Suppose that z is hidden by some vertex set $U' \neq \{z\}$. This U' is minimal without loss of generality. In particular, we can assume that U' is tight in the sense of Definition 6.25 and that $U' = U'\llbracket z \rrbracket$. Then by Corollary 6.33 it holds that U' is hiding-connected. Letting $L = \operatorname{minlevel}\bigl(U'\bigr)$ and setting $j = h$ in the spreading inequality (6.10), we get that $\bigl|U'\bigr| \geq 1 + h - L$ and hence $m\bigl(U'\bigr) \geq m^L\bigl(U'\bigr) \geq L + 2(1 + h - L) = 2h - L + 2 > h + 2$ since $L < h$. □

Proof of Theorem 9.24. The statement in the theorem follows from combining Observations 9.25 and 9.26 with the inequality (9.41). Hence, just as for Theorem 6.24 the crux of the matter is the induction proof needed to get this inequality.

Suppose that U_t is such that it blocks $\mathbb{S}_t$ and $\mathrm{pot}(\mathbb{S}_t) = m(U_t)$. By the inductive hypothesis, we have that $\mathrm{pot}(\mathbb{S}_t) \leq (2C_K + 1) \cdot \max_{s \leq t}\{\mathit{cost}(\mathbb{S}_s)\}$. We want to show for $\mathbb{S}_{t+1}$ that $\mathrm{pot}(\mathbb{S}_{t+1}) \leq (2C_K + 1) \cdot \max_{s \leq t+1}\{\mathit{cost}(\mathbb{S}_s)\}$. Clearly, this follows if we can prove that

$$\mathrm{pot}(\mathbb{S}_{t+1}) \leq \max\{\mathrm{pot}(\mathbb{S}_t), (2C_K + 1) \cdot \mathit{cost}(\mathbb{S}_t)\} \ . \tag{9.42}$$

We also note that if U_t blocks $\mathbb{S}_{t+1}$ we are done, since if so $\mathrm{pot}(\mathbb{S}_{t+1}) \leq m(U_t) = \mathrm{pot}(\mathbb{S}_t)$.

We make a case analysis depending on the type of move in Definition 9.7 made to get from $\mathbb{S}_t$ to $\mathbb{S}_{t+1}$. Analogously with the proof of Lemma 6.22, we want to show that we can use U_t to block $\mathbb{S}_{t+1}$ as long as the move is not an introduction on a source vertex and then use the Generalized LHC property to take care of such black pebble placements on sources.

Erasure $\mathbb{S}_{t+1} = \mathbb{S}_t \setminus \big\{[B]\langle W\rangle\big\}$ for $[B]\langle W\rangle \in \mathbb{S}_t$. Obviously, U_t blocks $\mathbb{S}_{t+1} \subseteq \mathbb{S}_t$.

Inflation $\mathbb{S}_{t+1} = \mathbb{S}_t \cup \big\{[B]\langle W\rangle\big\}$ for $[B]\langle W\rangle$ inflated from some $[B']\langle W'\rangle \in \mathbb{S}_t$ such that

$$B' \subseteq B \ , \tag{9.43a}$$

$$W' \cap \mathit{lpp}(B) \subseteq W \ , \text{ and} \tag{9.43b}$$

$$B \cap W' = \emptyset \ . \tag{9.43c}$$

We claim that U_t blocks $[B]\langle W\rangle$ and thus all of $\mathbb{S}_{t+1}$. Let us first argue intuitively why. Suppose that P is any source path agreeing with B. This path also agrees with B', and so must be blocked by $U_t \cup W'$ by assumption. If U_t blocks B we are done. We can worry, though, that U_t does not block P, but that instead P was blocked by some $w \in W'$ that disappeared as a result of the inflation move. But if $w \in W'$ is on a path via B, it cannot have disappeared, so this can never happen.

We now write down the formal details. With the notation in Definition 9.18, fix any path $P \in \mathrm{unblocked}([B]\langle W\rangle)$. Note that $P \cap W = \emptyset$ by definition. We need to show that $P \cap U_t \neq \emptyset$. Let us assume without loss of generality that P ends in $\mathrm{top}(B)$, for U_t blocks $[B]\langle W\rangle$ precisely if it blocks the paths $P \cap G_{\Delta}^{\mathrm{top}(B)}$ for all $P \in \mathrm{unblocked}([B]\langle W\rangle)$. We note that by definition, the fact that P agrees with a chain V and ends in $\mathrm{top}(V)$ implies that

$$P \subseteq V \,\dot{\cup}\, \mathit{lpp}(V) \ . \tag{9.44}$$

Since P agrees with B, or in formal notation $P \in \mathfrak{P}_{\mathrm{via}}(B)$, and since $B' \subseteq B$ by (9.43a), we have $P \in \mathfrak{P}_{\mathrm{via}}(B')$. By assumption, U_t blocks $[B']\langle W'\rangle$, which in particular means that $U_t \cup W'$ intersects the path P agreeing with B'. We get

$$\begin{aligned}
\emptyset \neq P \cap \bigl(U_t \cup W'\bigr) \qquad & \bigl[\text{ by definition of blocking }\bigr] \\
= (P \cap U_t) \cup \bigl((P \setminus B) \cap W'\bigr) \qquad & \bigl[\text{ since } B \cap W' = \emptyset \text{ by (9.43c) }\bigr] \\
= (P \cap U_t) \cup \bigl(P \cap \mathit{lpp}(B) \cap W'\bigr) \qquad & \bigl[\text{ since } P \subseteq B \mathbin{\dot\cup} \mathit{lpp}(B) \text{ by (9.44) }\bigr] \\
\subseteq (P \cap U_t) \cup (P \cap W) \qquad & \bigl[\ \mathit{lpp}(B) \cap W' \subseteq W \text{ by (9.43b) }\bigr] \\
= P \cap U_t \qquad & \bigl[\text{ since } P \cap W = \emptyset\ \bigr]
\end{aligned}$$

so $P \cap U_t \neq \emptyset$ and the desired conclusion that U_t blocks the path P follows.

Merger $\mathbb{S}_{t+1} = \mathbb{S}_t \cup \bigl\{[B]\langle W\rangle\bigr\}$ for $[B]\langle W\rangle$ derived by merger of two subconfigurations $[B_1]\langle W_1\rangle, [B_2]\langle W_2\rangle \in \mathbb{S}_t$ such that

$$B_1 \cap W_2 = \emptyset \ , \tag{9.45a}$$

$$B_2 \cap W_1 = \{v^*\} \ , \tag{9.45b}$$

$$B = (B_1 \cup B_2) \setminus \{v^*\} \ , \text{ and} \tag{9.45c}$$

$$W = \bigl((W_1 \cup W_2) \setminus \{v^*\}\bigr) \cap \mathit{lpp}(B) \ . \tag{9.45d}$$

Let us again first argue informally that if a set of vertices U_t blocks two subconfigurations $[B_1]\langle W_1\rangle$ and $[B_2]\langle W_2\rangle$, it must also block their merger. Let P be any path via B, and suppose in addition that P visits the merger vertex v^*. If so, P agrees with B_2 and must be blocked by $U_t \cup W_2$. If on the other hand P agrees with B but does *not* visit v^*, it is a path via B_1 that in addition does not pass through the white pebble in W_1 eliminated in the merger. This means that $U_t \cup W_1 \setminus \{v^*\}$ must block P. Again, we have to argue that the blocking white vertices do not disappear when we apply the intersection with $\mathit{lpp}(B)$ in (9.45d), but this is straightforward to verify.

So let us show formally that U_t blocks $[B]\langle W\rangle$, i.e., that it must hold for any path $P \in \text{unblocked}([B]\langle W\rangle)$ that $P \cap U_t \neq \emptyset$. As above, without loss of generality we consider only paths P ending in $\text{top}(B) = \text{top}(B_1 \cup B_2)$. Recall that

$$B_i \cap W_i = \emptyset \tag{9.46}$$

holds for all subconfigurations by definition. Also, the set inclusion

$$\mathit{lpp}(B \cup \{v^*\}) \subseteq \mathit{lpp}(B) \setminus \{v^*\} \tag{9.47}$$

is easy to verify. We divide the analysis into two subcases.

1. $P \in \mathfrak{P}_{\mathrm{via}}(B_1 \cup B_2) = \mathfrak{P}_{\mathrm{via}}(B \cup \{v^*\})$. If so, in particular it holds that $P \in \mathfrak{P}_{\mathrm{via}}(B_2)$ and since U_t blocks $[B_2]\langle W_2\rangle$ we have

$$\begin{aligned} \emptyset \neq P \cap \big(U_t \cup W_2\big) & & \big[\text{ by definition }\big] \\ = (P \cap U_t) \cup \big((P \setminus (B_1 \cup B_2)) \cap W_2\big) & & \big[\text{ by (9.45a) \& (9.46) }\big] \\ = (P \cap U_t) \cup \big(P \cap \mathit{lpp}(B_1 \cup B_2) \cap W_2\big) & & \big[\text{ by (9.44) }\big] \\ = (P \cap U_t) \cup \big(P \cap \mathit{lpp}(B \cup v^*) \cap W_2\big) & & \big[\text{ by (9.45c) }\big] \\ \subseteq (P \cap U_t) \cup \big(P \cap (W_2 \setminus \{v^*\}) \cap \mathit{lpp}(B) & & \big[\text{ by (9.47) }\big] \\ \subseteq (P \cap U_t) \cup (P \cap W) & & \big[\text{ by (9.45d) }\big] \\ = P \cap U_t & & \big[\text{ since } P \cap W = \emptyset\ \big] \end{aligned}$$

so U_t blocks the path P in this case.

2. $P \in \mathfrak{P}_{\mathrm{via}}(B) \setminus \mathfrak{P}_{\mathrm{via}}(B \cup \{v^*\})$. This means that $B \subseteq P$ but $B \cup \{v^*\} \nsubseteq P$, so the path P does not pass through v^*. Since P agrees with B_1 and U_t blocks $[B_1]\langle W_1\rangle$ by assumption, we get that

$$\begin{aligned} \emptyset \neq P \cap \big(U_t \cup W_1\big) & & \big[\text{ by definition }\big] \\ = (P \cap U_t) \cup \big((P \setminus B) \cap W_1\big) & & \big[\text{ (9.45b) \& (9.46) }\big] \\ = (P \cap U_t) \cup \big(P \cap \mathit{lpp}(B) \cap W_1\big) & & \big[\text{ by (9.44) }\big] \\ = (P \cap U_t) \cup \big(P \cap (W_1 \setminus \{v^*\}) \cap \mathit{lpp}(B)\big) & & \big[\text{ since } v^* \notin P\ \big] \\ \subseteq (P \cap U_t) \cup (P \cap W) & & \big[\text{ by (9.45d) }\big] \\ = (P \cap U_t) & & \big[\text{ since } P \cap W = \emptyset\ \big] \end{aligned}$$

and U_t blocks the path P in this case as well.

Introduction $\mathbb{S}_{t+1} = \mathbb{S}_t \cup \big\{[v]\langle \mathit{pred}(v)\rangle\big\}$. Clearly, U_t blocks $\mathbb{S}_{t+1}$ if v is a non-source vertex, i.e., if $\mathit{pred}(v) \neq \emptyset$, since U_t blocks $\mathbb{S}_t$ and $[v]\langle \mathit{pred}(v)\rangle$ blocks itself.

Suppose however that v is a source vertex, so that the subconfiguration introduced is $[v]\langle\emptyset\rangle$. As in the proof of Lemma 6.22, U_t does not necessarily block $\mathbb{S}_{t+1}$ any longer but $U_{t+1} = U_t \cup \{v\}$ clearly does. For $j > 0$, it holds that $U_{t+1}\{\succeq j\} = U_t\{\succeq j\}$ and thus $m^j(U_{t+1}) = m^j(U_t)$. On the bottom level $j = 0$, using that $|U_t| \leq C_K \cdot \mathsf{cost}(\mathbb{S}_t)$ Generalized LHC property 9.23 we have

$$\begin{aligned} m^0(U_{t+1}) = 2 \cdot |U_{t+1}| = 2 \cdot (|U_t| + 1) \leq \\ 2 \cdot \big(C_K \cdot \mathsf{cost}(\mathbb{S}_t) + 1\big) \leq 2 \cdot \big(C_K \cdot \mathsf{cost}(\mathbb{S}_{t+1}) + 1\big) \leq \\ 2 \cdot \big(C_K \cdot \mathsf{cost}(\mathbb{S}_{t+1}) + \mathsf{cost}(\mathbb{S}_{t+1})\big) \leq 2(C_K + 1) \cdot \mathsf{cost}(\mathbb{S}_{t+1}) \end{aligned} \tag{9.48}$$

and we get that

$$\begin{aligned}\mathrm{pot}(\mathbb{S}_{t+1}) \leq m(U_{t+1}) \leq \max_j\bigl\{m^j(U_{t+1})\bigr\}\\ \leq \max\bigl\{m(U_t), (2C_K+1)\cdot \mathit{cost}(\mathbb{S}_{t+1})\bigr\} =\\ \max\bigl\{\mathrm{pot}(\mathbb{S}_t), (2C_K+1)\cdot \mathit{cost}(\mathbb{S}_{t+1})\bigr\} \quad (9.49)\end{aligned}$$

which is what is needed for the induction step to go through.

We see that regardless of the pebbling move made in the transition $\mathbb{S}_t \rightsquigarrow \mathbb{S}_{t+1}$, the inequality (9.42) holds. The theorem follows by the induction principle. □

Hence, in order to prove a lower bound on *Blob-Peb*(G_h) for layered spreading graphs G_h, it is sufficient to find some constant C_K such that these DAGs can be shown to possess the Generalized LHC property 9.23 with parameter C_K.

9.5.3 Some Structural Transformations

As we tried to indicate by presenting the small toy blob-pebbling configurations in Examples 9.20 and 9.21, the potential in the blob-pebble game behaves somewhat differently from the potential in the standard pebble game. There are (at least) two important differences:

- Firstly, for the white pebbles we have to keep track of exactly which black pebbles they can help to block. This can lead to slightly unexpected consequences such as the blocking set U and the set of white pebbles overlapping.
- Secondly, for black blobs there is a much wider choice where to block the blob-pebbles than for atomic pebbles. It seems that to minimize the potential, blocking black blobs on (reasonably) low levels should still be a good idea. However, we cannot a priori exclude the possibility that if a lot of black blobs intersect in some high-level vertex, adding this vertex to a blocking set U might be a better idea.

In this subsection we address the first of these issues. The second issue, which turns out to be much trickier, is dealt with in the next subsection.

One simplifying observation is that we do not have to prove Property 9.23 for arbitrary blob-pebbling configurations. Below, we show that one can do some technical preprocessing of the blob-pebbling configurations so that it suffices to prove the Generalized LHC property for the subclass of configurations resulting from this preprocessing.[1] Throughout this subsection, we assume that the parameter C_K is some fixed constant.

[1] Note that we did something similar in Section 6.5 after Lemma 6.28, when we argued that if U is a minimum-measure hiding set for $\mathbb{P} = (B, W)$, we can assume without loss of generality that $U \cup W$ is tight. For if not, we just prove the Limited hiding-cardinality property for some tight subset $U' \cup W' \subseteq U \cup W$ instead. This is wholly analogous to the reasoning here, but since matters become more complex we need to be a bit more careful.

We start slowly by taking care of a pretty obvious redundancy. Let us say that the blob subconfiguration $[B]\langle W\rangle$ is *self-blocking* if W blocks B. The blob-pebbling configuration $\mathbb{S}$ is *self-blocker-free* if there are no self-blocking subconfigurations in $\mathbb{S}$. That is, if $[B]\langle W\rangle$ is self-blocking, W needs no extra help blocking B. Perhaps the simplest example of this is $[B]\langle W\rangle = [v]\langle pred(v)\rangle$ for a non-source vertex v. The following proposition is immediate.

Proposition 9.27. *For $\mathbb{S}$ any blob-pebbling configuration, let $\mathbb{S}'$ be the blob-pebbling configuration with all self-blockers in $\mathbb{S}$ removed. Then $\mathsf{cost}(\mathbb{S}') \leq \mathsf{cost}(\mathbb{S})$, $\mathrm{pot}(\mathbb{S}') = \mathrm{pot}(\mathbb{S})$ and any blocking set U' for $\mathbb{S}'$ is also a blocking set for $\mathbb{S}$.*

Corollary 9.28. *Suppose the Generalized LHC property holds for self-blocker-free blob-pebbling configurations. Then the Generalized LHC property holds for all blob-pebbling configurations.*

Proof. If $\mathbb{S}$ is *not* self-blocker-free, take the maximal $\mathbb{S}' \subseteq \mathbb{S}$ that is and the blocking set U' that the Generalized LHC property provides for this $\mathbb{S}'$. Then U' blocks $\mathbb{S}$ and since the two configurations $\mathbb{S}$ and $\mathbb{S}'$ have the same blocking sets their potentials are equal, so $\mathrm{pot}(\mathbb{S}) = m(U')$. Finally, we have that $|U| \leq C_K \cdot \mathsf{cost}(\mathbb{S}') \leq C_K \cdot \mathsf{cost}(\mathbb{S})$. Thus the Generalized LHC property holds for $\mathbb{S}$. □

We now move on to a more interesting observation. Looking at the blob-pebbling configuration $\mathbb{S} = \{[z]\langle y_1\rangle, [z]\langle y_2\rangle\}$ in Example 9.20, it seems that the white pebbles really do not help at all. One might ask if we could not just throw them away? Perhaps somewhat surprisingly, the answer is yes, and we can capture the intuitive concept of necessary white pebbles and formalize it as follows.

Definition 9.29 (White sharpening). Given $\mathbb{S} = \{[B_i]\langle W_i\rangle\}_{i\in[m]}$, we say that $\mathbb{S}'$ is a *white sharpening* of $\mathbb{S}$ if $\mathbb{S}' = \{[B'_i]\langle W'_i\rangle\}_{i\in[m]}$ for $B'_i = B_i$ and $W'_i \subseteq W_i$.

That is, a white sharpening removes white pebbles and thus makes the blob-pebbling configuration stronger or "sharper" in the sense that the cost can only decrease and the potential can only increase.

Proposition 9.30. *If $\mathbb{S}'$ is a white sharpening of $\mathbb{S}$ it holds that $\mathsf{cost}(\mathbb{S}') \leq \mathsf{cost}(\mathbb{S})$ and $\mathrm{pot}(\mathbb{S}') \geq \mathrm{pot}(\mathbb{S})$. More precisely, any blocking set U' for $\mathbb{S}'$ is also a blocking set for $\mathbb{S}$.*

Proof. The statement about cost is immediate from Definition 9.8. The statement about potential clearly follows from Definition 9.19 since it holds that any blocking set U' for $\mathbb{S}'$ is also a blocking set for $\mathbb{S}$. □

In the next definition, we suppose that there is some fixed but arbitrary ordering of the vertices in G, and that the vertices are considered in this order.

Definition 9.31 (White elimination). For $[B]\langle W\rangle$ a subconfiguration and U any blocking set for $[B]\langle W\rangle$, write $W = \{w_1, \ldots, w_s\}$, set $W^0 := W$ and iteratively perform the following for $i = 1, \ldots, s$: If $U \cup (W^{i-1} \setminus \{w_i\})$ blocks B, set $W^i := W^{i-1} \setminus \{w_i\}$, otherwise set $W^i := W^{i-1}$. We define the *white elimination* of $[B]\langle W\rangle$ with respect to U to be $\mathcal{W}\text{-elim}([B]\langle W\rangle, U) = [B]\langle W^s\rangle$ for W^s the final set resulting from the procedure above.

For $\mathbb{S}$ a blob-pebbling configuration and U a blocking set for $\mathbb{S}$, we define

$$\mathcal{W}\text{-elim}(\mathbb{S}, U) = \left\{\mathcal{W}\text{-elim}([B]\langle W\rangle, U) \,\middle|\, [B]\langle W\rangle \in \mathbb{S}\right\} . \tag{9.50}$$

We say that the elimination is *strict* if $\mathbb{S} \neq \mathcal{W}\text{-elim}(\mathbb{S}, U)$. If $\mathbb{S} = \mathcal{W}\text{-elim}(\mathbb{S}, U)$ we say that $\mathbb{S}$ is *white-eliminated*, or *$\mathcal{W}$-eliminated* for short, with respect to U.

Clearly $\mathcal{W}\text{-elim}(\mathbb{S}, U)$ is a white sharpening of $\mathbb{S}$. And if we pick the right U, we simplify the problem of proving the Generalized LHC property a bit more.

Lemma 9.32. *If U is a minimum-measure blocking set for $\mathbb{S}$, then it holds that $\mathbb{S}' = \mathcal{W}\text{-elim}(\mathbb{S}, U)$ is a white sharpening of $\mathbb{S}$ such that $\mathrm{pot}(\mathbb{S}') = \mathrm{pot}(\mathbb{S})$ and U blocks $\mathbb{S}'$.*

Proof. Since $\mathbb{S}' = \mathcal{W}\text{-elim}(\mathbb{S}, U)$ is a white sharpening of $\mathbb{S}$ (which is easily verified from Definitions 9.29 and 9.31), it holds by Proposition 9.30 that $\mathrm{pot}(\mathbb{S}') \geq \mathrm{pot}(\mathbb{S})$. Looking at the construction in Definition 9.31, we also see that the white pebbles are "sharpened away" with care so that U remains a blocking set. Thus $m(U) \geq \mathrm{pot}(\mathbb{S}') = \mathrm{pot}(\mathbb{S}) = m(U)$, and the lemma follows. □

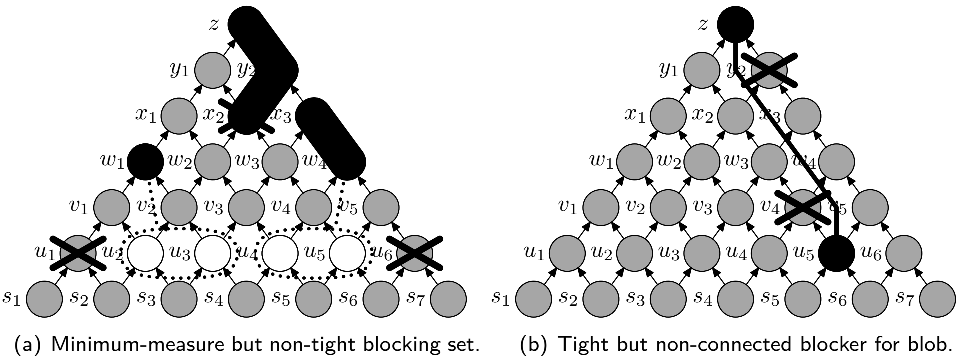
Corollary 9.33. *Suppose that the Generalized LHC property holds for the set of all blob-pebbling configurations $\mathbb{S}$ having the property that for all minimum-measure blocking sets U for $\mathbb{S}$ it holds that $\mathbb{S} = \mathcal{W}\text{-elim}(\mathbb{S}, U)$. Then the Generalized LHC property holds for all blob-pebbling configurations.*

Proof. This is essentially the same reasoning as in the proof of Corollary 9.28 plus induction. Let $\mathbb{S}$ be any blob-pebbling configuration. Suppose that there exists a minimum-measure blocker U for $\mathbb{S}$ such that $\mathbb{S}$ is not $\mathcal{W}$-eliminated with respect to U. Let $\mathbb{S}^1 = \mathcal{W}\text{-elim}(\mathbb{S}, U)$. Then $\mathsf{cost}(\mathbb{S}^1) \leq \mathsf{cost}(\mathbb{S})$ by Proposition 9.30 and $\mathrm{pot}(\mathbb{S}^1) = \mathrm{pot}(\mathbb{S})$ by Lemma 9.32.

If there is a minimum-measure blocker U^1 for $\mathbb{S}^1$ such that $\mathbb{S}^1$ is not $\mathcal{W}$-eliminated with respect to U^1, set $\mathbb{S}^2 = \mathcal{W}\text{-elim}(\mathbb{S}^1, U^1)$. Continuing in this manner, we get a chain $\mathbb{S}^1, \mathbb{S}^2, \mathbb{S}^3, \ldots$ of strict $\mathcal{W}$-eliminations such that $\mathsf{cost}(\mathbb{S}^1) \geq \mathsf{cost}(\mathbb{S}^2) \geq \mathsf{cost}(\mathbb{S}^3) \ldots$ and $\mathrm{pot}(\mathbb{S}^1) = \mathrm{pot}(\mathbb{S}^2) = \mathrm{pot}(\mathbb{S}^3) = \ldots$ This chain must terminate at some configuration $\mathbb{S}^k$ since the total number of white pebbles (counted with repetitions) decreases in every round.

Let U^k be the blocker that the Generalized LHC property provides for $\mathbb{S}^k$. Then U^k blocks $\mathbb{S}$, $\mathrm{pot}(\mathbb{S}) = \mathrm{pot}(\mathbb{S}^k) = m(U^k)$, and $|U^k| \leq C_K \cdot \mathsf{cost}(\mathbb{S}^k) \leq C_K \cdot \mathsf{cost}(\mathbb{S})$. Thus the Generalized LHC property holds for $\mathbb{S}$. □

We note that in particular, it follows from the construction in Definition 9.31 combined with Corollary 9.33 that we can assume without loss of generality for any blocking set U and any blob-pebbling configuration $\mathbb{S}$ that U does not intersect the set of white-pebbled vertices in $\mathbb{S}$.



(a) Minimum-measure but non-tight blocking set.

(b) Tight but non-connected blocker for blob.

Figure 9.4: Two blob-pebbling configurations with problematic blocking sets.

Proposition 9.34. *If* $\mathbb{S} = \mathcal{W}\text{-elim}(\mathbb{S}, U)$*, then it holds that* $U \cap \mathcal{W}(\mathbb{S}) = \emptyset$*.*

Proof. Any $w \in \mathcal{W}(\mathbb{S}) \cap U$ would have been removed in the $\mathcal{W}$-elimination. □

9.5.4 A Proof of the Generalized Limited Hiding-Cardinality Property

We are now ready to embark on the proof of the Generalized LHC property for layered spreading DAGs.

Theorem 9.35. *All layered blob-pebblable DAGs that are spreading possess the Generalized limited hiding-cardinality property 9.23 with parameter* $C_K = 13$*.*

Since pyramids are spreading graphs by Theorem 6.45, this is all that we need to get the lower bound on blob-pebbling price on pyramids from Theorem 9.24. We note that the parameter C_K in Theorem 9.35 can easily be improved. However, our main concern here is not optimality of constants but clarity of exposition.

We prove Theorem 9.35 by applying the preprocessing in the previous subsection and then (almost) reducing the problem to the standard black-white pebble game. However, some twists are added along the way since our potential measure for blobs behave differently from Klawe's potential measure for black and white pebbles. Let us first exemplify two problems that arise if we try to do naive pattern matching on Klawe's proof for the standard black-white pebble game.

In the standard black-white pebble game, if U is a minimum-measure hiding set for $\mathbb{P} = (B, W)$, Lemma 6.28 tells us that we can assume without loss of generality that $U \cup W$ is tight. This is *not* true in the blob-pebble game, not even after the transformations in Section 9.5.3.

Example 9.36. Consider $\mathbb{S} = \{[w_1]\langle u_2, u_3\rangle, [w_4, x_3]\langle u_4, u_5\rangle, [x_2, y_2, z]\langle\emptyset\rangle\}$ in Figure 9.4(a) with blocking set $U = \{x_2, u_1, u_6\}$. It can be verified that U is a minimum-measure blocking set and that the configuration $\mathbb{S}$ is $\mathcal{W}$-eliminated with respect to U, but the set $U \cup \mathcal{W}(\mathbb{S}) = \{u_1, u_2, u_3, u_4, u_5, u_6, x_2\}$ is not tight (because of x_2).

This can be handled, but a more serious problem is that even if the set $U \cup W$ blocking the chain B is tight, there is no guarantee that the vertices in $U \cup W$ end up in the same connected component of the hiding set graph $\mathcal{H}(U \cup W)$ in Definition 6.30.

Example 9.37. Consider the single-blob configuration $\mathbb{S} = \{[u_5, z]\langle\emptyset\rangle\}$ drawn in Figure 9.4(b). It is easy to verify that $U = \{v_4, y_2\}$ is a subset-minimal blocker of $\mathbb{S}$ and also a tight vertex set. This highlights the fact that blocking sets for blob-pebbling configurations can have rather different properties than hiding sets for standard pebbles. In particular, a minimal blocking set for a single blob can have several "isolated" vertices at large distances from one another. Among other problems, this leads to difficulties in defining connected components of blocking sets for subconfigurations.

The naive attempt to generalize Definition 6.30 of connected components in a hiding set graph to blocking sets would place the vertices v_4 and y_2 in different connected components $\{v_4\}$ and $\{y_2\}$, none of which blocks $\mathbb{S} = \{[u_5, z]\langle\emptyset\rangle\}$. This is not what we want (compare Corollary 6.33 for hiding sets for black-white pebble configurations). We remark that there really cannot be any other sensible definition that places v_4 and y_2 in the same connected component either, at least not if we want to appeal to the spreading properties in Definition 6.44. Since the level difference in U is 3 but the size of the set is only 2, the spreading inequality (6.10) cannot hold for this set.

To get around this problem, we will instead use connected components defined in terms of hiding the singleton black pebbles given by the bottom vertices of our blobs. For a start, recalling Definitions 6.16 and 9.17, let us make an easy observation relating the hiding and blocking relations for a blob.

Observation 9.38. *If a vertex set V hides some vertex $b \in B$, then V blocks B.*

Proof. If V blocks all paths visiting b, then in particular it blocks the subset of paths that not only visits b but agree with all of B. □

We will focus on the case when the bottom vertex of a blob is hidden.

Definition 9.39 (Hiding blob-pebbling configurations). We say that the vertex set U *hides* the subconfiguration $[B]\langle W\rangle$ if $U \cup W$ hides the vertex $\mathrm{bot}(B)$, and that U hides the blob-pebbling configuration $\mathbb{S}$ if U hides all $[B]\langle W\rangle \in \mathbb{S}$.

If U does not hide $[B]\langle W\rangle$, then U blocks $[B]\langle W\rangle$ only if $U \cap G^{\triangledown}_{\mathrm{bot}(B)}$ does.

Proposition 9.40. *Suppose that a vertex set U in a layered DAG G blocks but does not hide the subconfiguration $[B]\langle W\rangle$ and that $[B]\langle W\rangle$ does not block itself. Then $U \cap G_{\vartriangle}^{\mathrm{bot}(B)}$ does not block $[B]\langle W\rangle$, but there is a subset $U' \subseteq U \cap G^{\triangledown}_{\mathrm{bot}(B)}$ that blocks $[B]\langle W\rangle$.*

Proof. Suppose that $U \cup W$ blocks B but does not hide $b = \mathrm{bot}(B)$, and that W does not block B. Then there is a source path P_2 via B such that $P_2 \cap W = \emptyset$. Also, there is a source path P_1 to b such that $P_1 \cap (U \cup W) = \emptyset$. Let $P = \bigl(P_1 \cap G^b_\vartriangle\bigr) \cup \bigl(P_2 \cap G^\triangledown_b\bigr)$ be the source path that starts like P_1 and continues like P_2 from b onwards. Clearly,

$$P \cap \bigl((U \cap G^b_\vartriangle) \cup W\bigr) = \bigl(P_1 \cap (U \cup W)\bigr) \cup \bigl(P_2 \cap W\bigr) = \emptyset \tag{9.51}$$

so $U \cap G^b_\vartriangle$ does not block $[B]\langle W\rangle$.

Suppose that $U \cap G^\triangledown_b$ does not block $[B]\langle W\rangle$. Since $U \cup W$ does not hide b, there is some source path P_1 to b with $P_1 \cap (U \cup W) = \emptyset$. Also, since $U \cup W$ blocks B but $\bigl(U \cap G^\triangledown_b\bigr) \cup W$ does not, there is a source path P_2 via B such that $P_2 \cap (U \cup W) \neq \emptyset$ but $P_2 \cap (U \cup W) \cap G^\triangledown_b = \emptyset$. But then let $P = \bigl(P_1 \cap G^b_\vartriangle\bigr) \cup \bigl(P_2 \cap G^\triangledown_b\bigr)$ be the source path that starts like P_1 and continues like P_2 from b onwards. We get that P agrees with B and that $P \cap (U \cup W) = \emptyset$, contradicting the assumption that U blocks $[B]\langle W\rangle$. □

We want to distinguish between subconfigurations that are hidden and subconfigurations that are just blocked, but not hidden. To this end, let us introduce the notation

$$\mathbb{S}_H(\mathbb{S}, U) = \bigl\{[B]\langle W\rangle \in \mathbb{S} \bigm| U \text{ hides } [B]\langle W\rangle\bigr\} \tag{9.52}$$

to denote the subconfigurations in $\mathbb{S}$ hidden by U and

$$\mathbb{S}_B(\mathbb{S}, U) = \mathbb{S} \setminus \mathbb{S}_H(\mathbb{S}, U) \tag{9.53}$$

to denote the subconfigurations that are just blocked. We write

$$\mathcal{B}_H(\mathbb{S}, U) = \{\mathrm{bot}(B) \mid [B]\langle W\rangle \in \mathbb{S}_H(\mathbb{S}, U)\} \tag{9.54}$$

$$\mathcal{B}_B(\mathbb{S}, U) = \{\mathrm{bot}(B) \mid [B]\langle W\rangle \in \mathbb{S}_B(\mathbb{S}, U)\} \tag{9.55}$$

to denote the black bottom vertices in these two subsets of subconfigurations and note that we can have $\mathcal{B}_H(\mathbb{S}, U) \cap \mathcal{B}_B(\mathbb{S}, U) \neq \emptyset$. The white pebbles in these subsets located below the bottom vertices of the black blobs that they are supporting are denoted

$$\mathcal{W}^\vartriangle_H(\mathbb{S}, U) = \bigl\{W \cap G^b_\vartriangle \bigm| [B]\langle W\rangle \in \mathbb{S}_H(\mathbb{S}, U),\, b = \mathrm{bot}(B)\bigr\} \tag{9.56}$$

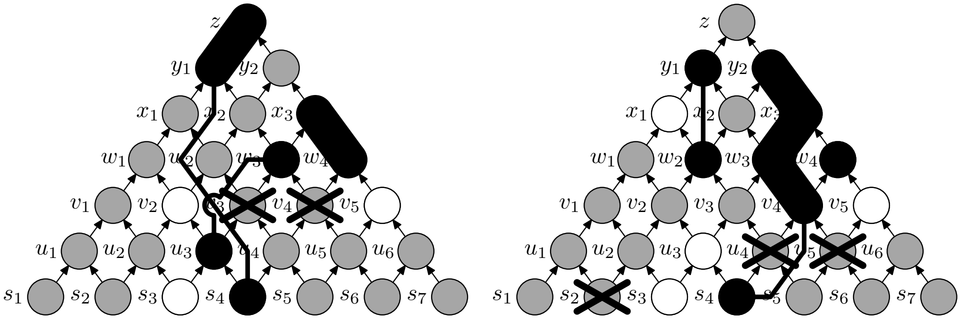
and

$$\mathcal{W}^\vartriangle_B(\mathbb{S}, U) = \bigl\{W \cap G^b_\vartriangle \bigm| [B]\langle W\rangle \in \mathbb{S}_B(\mathbb{S}, U),\, b = \mathrm{bot}(B)\bigr\} \ . \tag{9.57}$$

This notation will be used heavily in what follows, so we give a couple of simple but hopefully illuminating examples before we continue.

Example 9.41. Consider the blob-pebbling configurations and blocking sets in Figure 9.5. For the blob-pebbling configuration

$$\mathbb{S}_1 = \bigl\{[s_4, y_1, z]\langle v_2\rangle, [u_3, w_3]\langle s_3\rangle, [w_4, x_3]\langle v_5\rangle\bigr\} \tag{9.58}$$



(a) $\mathbb{S}_1$ in Example 9.41.

(b) $\mathbb{S}_2$ in Example 9.41.

Figure 9.5: Blob-pebbling configurations with hidden and just blocked blobs.

with blocking set $U_1 = \{v_3, v_4\}$ in Figure 9.5(a), the vertex set $\{v_4, v_5\}$ hides $w_4 = \text{bot}([w_4, x_3])$ but $[s_4, y_1, z]$ is blocked but not hidden by $\{v_2, v_3, v_4\}$ and $[u_3, w_3]$ is blocked but not hidden by $\{v_3\}$. Thus, we have

$$\begin{aligned}
\mathbb{S}_H(\mathbb{S}_1, U_1) &= \big\{[w_4, x_3]\langle v_5\rangle\big\} \\
\mathbb{S}_B(\mathbb{S}_1, U_1) &= \big\{[s_4, y_1, z]\langle v_2\rangle, [u_3, w_3]\langle s_3\rangle\big\} \\
\mathcal{B}_H(\mathbb{S}_1, U_1) &= \{w_4\} \\
\mathcal{B}_B(\mathbb{S}_1, U_1) &= \{s_4, u_3\} \\
\mathcal{W}_H^{\triangle}(\mathbb{S}_1, U_1) &= \{v_5\} \\
\mathcal{W}_B^{\triangle}(\mathbb{S}_1, U_1) &= \{s_3\}
\end{aligned}$$

in this example. For the configuration

$$\mathbb{S}_2 = \big\{[s_4, v_4, w_3, x_3, y_2]\langle\emptyset\rangle, [w_2, y_1]\langle s_3, u_3, x_1\rangle, [w_4]\langle v_5\rangle\big\} \tag{9.59}$$

with blocker $U_2 = \{s_2, u_4, u_5\}$ in Figure 9.5(b), it is straightforward to verify that

$$\begin{aligned}
\mathbb{S}_H(\mathbb{S}_2, U_2) &= \big\{[w_2, y_1]\langle s_3, u_3, x_1\rangle, [w_4]\langle v_5\rangle\big\} \\
\mathbb{S}_B(\mathbb{S}_2, U_2) &= \big\{[s_4, v_4, w_3, x_3, y_2]\langle\emptyset\rangle\big\} \\
\mathcal{B}_H(\mathbb{S}_2, U_2) &= \{w_2, w_4\} \\
\mathcal{B}_B(\mathbb{S}_2, U_2) &= \{s_4\} \\
\mathcal{W}_H^{\triangle}(\mathbb{S}_2, U_2) &= \{s_3, u_3, v_5\} \\
\mathcal{W}_B^{\triangle}(\mathbb{S}_2, U_2) &= \emptyset
\end{aligned}$$

are the corresponding sets.

Let us also use the opportunity to illustrate Definition 9.31. The blob-pebbling configuration $\mathbb{S}_1$ is not $\mathcal{W}$-eliminated with respect to U_1, since U_1 also blocks this configuration with the white pebble on s_3 removed. However, a better idea measure-wise is to change the blocking set for $\mathbb{S}_1$ to $U_1' = \{s_4, v_4\}$, which has measure $m(U_1') = 4 < 6 = m(U_1)$. The vertex set U_2 can be verified to be a minimum-measure blocker for $\mathbb{S}_2$, but when $\mathbb{S}_2$ is $\mathcal{W}$-eliminated with respect to U_2 the white pebble on x_1 disappears.

As a final remark in this example, we comment that although we have not indicated explicitly in Figures 9.5(a) and 9.5(b) which white pebbles W are associated with which black blob B (as was done in Figure 9.4(a)), this is uniquely determined by the requirement in Definition 9.6 that $W \subseteq lpp(B)$.

For the rest of this section we will assume without loss of generality (in view of Proposition 9.27 and Corollary 9.33) that we are dealing with a blob-pebbling configuration $\mathbb{S}$ and a minimum-measure blocker U of $\mathbb{S}$ such that $\mathbb{S}$ is free from self-blocking subconfigurations and is $\mathcal{W}$-eliminated with respect to U. As an aside, we note that it is not hard to show (using Definition 9.31 and Proposition 9.40) that this implies that $\mathcal{W}_B^{\triangle}(\mathbb{S}, U) = \emptyset$. We will tend to drop the arguments $\mathbb{S}$ and U for $\mathbb{S}_H, \mathbb{S}_B, \mathcal{B}_H, \mathcal{B}_B, \mathcal{W}_H^{\triangle}$, and $\mathcal{W}_B^{\triangle}$, since from now on the blob-pebbling configuration $\mathbb{S}$ and the blocker U will be fixed. With this notation, Theorem 9.35 clearly follows if we can prove the following lemma.

Lemma 9.42. *Let $\mathbb{S}$ be any blob-pebbling configuration on a layered spreading DAG and U be any blocking set for $\mathbb{S}$ such that*

1. $\mathrm{pot}(\mathbb{S}) = m(U)$, *i.e., U is a minimum-measure blocker of $\mathbb{S}$,*

2. *$\mathbb{S}$ is free from self-blocking subconfigurations and is $\mathcal{W}$-eliminated with respect to U, and*

3. *U has minimal size among all blocking sets U' for $\mathbb{S}$ such that* $\mathrm{pot}(\mathbb{S}) = m(U')$.

Then $|U| \leq 13 \cdot \left|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}_H^{\triangle}\right|$.

The proof is by contradiction, although we will have to work harder than for the corresponding Theorem 6.35 for black-white pebbling and also use (the proof of) the latter theorem as a subroutine. Thus, for the rest of this section, let us assume on the contrary that U has all the properties stated in Lemma 9.42 but that $|U| > 13 \cdot \left|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}_H^{\triangle}\right|$. We will show that this leads to a contradiction.

For the subconfiguration in $\mathbb{S}_H$ that are hidden by U, one could argue that matters should be reasonably similar to the case for standard black-white pebbling, and hopefully we could apply similar reasoning as in Section 6.5 to prove something useful about the vertex set hiding these subconfigurations. The subconfigurations in $\mathbb{S}_B$ that are just blocked but not hidden, however, seem harder to get a handle on (compare Example 9.37).

Let $U_H \subseteq U$ be a smallest vertex set hiding $\mathbb{S}_H$ and let $U_B = U \setminus U_H$. The set U_B consists of vertices that are not involved in any hiding of subconfigurations in $\mathbb{S}_H$, but only in blocking subconfigurations in $\mathbb{S}_B$ on levels above their bottom vertices. As a first step towards proving Lemma 9.42, and thus Theorem 9.35, we want to argue that U_B cannot be very large.

Consider the blobs in $\mathbb{S}_B$. By definition they are not hidden, but are blocked at some level above level(bot(B)). Since the vertices in U_B are located on high levels, a naive attempt to improve the blocking set would be to pick some vertex $u \in U_B$ and replace it by the vertices in $\mathcal{B}_B$ corresponding to the subconfigurations in $\mathbb{S}_B$ that u is involved in blocking, i.e., by the set

$$\mathcal{B}^u = \big\{\mathrm{bot}(B) \big| U \setminus \{u\} \text{ does not block } [B]\langle W\rangle \in \mathbb{S}_B\big\} \ . \tag{9.60}$$

Note that $\mathcal{B}^u$ is lower down in the graph than u, so $(U \setminus \{u\}) \cup \mathcal{B}^u$ is obtained from U by moving vertices downwards and by construction $(U \setminus \{u\}) \cup \mathcal{B}^u$ blocks $\mathbb{S}$. But by assumption, U has minimal potential and cardinality, so this new blocking set cannot be an improvement measure- or cardinality-wise. The same holds if we extend the construction to subsets $U' \subseteq U_B$ and the corresponding bottom vertices $\mathcal{B}^{U'} \subseteq \mathcal{B}_B$. By assumption we can never find any subset such that $(U \setminus \{U'\}) \cup \mathcal{B}^{U'}$ is a better blocker than U. It follows that the cost of the blobs that U_B helps to block must be larger than the size of U_B, and in particular that $|U_B| \leq |\mathcal{B}_B|$. Let us write this down as a lemma and prove it properly.

Lemma 9.43. *Let $\mathbb{S}$ be any blob-pebbling configuration on a layered DAG and U be any blocking set for $\mathbb{S}$ such that* $\mathrm{pot}(\mathbb{S}) = m(U)$*, U has minimal size among all blocking sets U' for $\mathbb{S}$ with* $\mathrm{pot}(\mathbb{S}) = m(U')$*, and $\mathbb{S}$ is free from self-blocking subconfigurations and is $\mathcal{W}$-eliminated with respect to U. Then if $U_H \subseteq U$ is any smallest set hiding $\mathbb{S}_H$ and $U_B = U \setminus U_H$, it holds that $|U_B| \leq |\mathcal{B}_B|$.*

Before proving this lemma, we note the immediate corollary that if the whole blocking set U is significantly larger than *cost*($\mathbb{S}$), the lion's share of U by necessity consists not of vertices blocking subconfigurations in $\mathbb{S}_B$, but of vertices hiding subconfigurations in $\mathbb{S}_H$. And recall that we are indeed assuming, to get a contradiction, that U is large.

Corollary 9.44. *Assume that $\mathbb{S}$ and U are as in Lemma 9.42 but with $|U| > 13 \cdot \big|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}_H^{\triangle}\big|$. Let $U_H \subseteq U$ be a smallest set hiding $\mathbb{S}_H$. Then it holds that $|U_H| > 12 \cdot \big|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}_H^{\triangle}\big|$.*

As was indicated in the informal discussion preceding Lemma 9.43, the proof of the lemma uses the easy observation that moving vertices downwards can only decrease the measure.

Observation 9.45. *Suppose that U, V_1 and V_2 are vertex sets in a layered DAG such that $U \cap V_2 = \emptyset$ and there is a one-to-one (but not necessarily onto) mapping $f : V_1 \mapsto V_2$ with the property that* $\mathrm{level}(v) \leq \mathrm{level}(f(v))$. *Then $m(U \cup V_1) \leq m(U \cup V_2)$.*

Proof. This follows immediately from Definition 6.18 on page 67 since the mapping f tells us that

$$\begin{aligned}|(U \cup V_1)\{\succeq j\}| \leq |U\{\succeq j\}| + |V_1\{\succeq j\}| &\leq |U\{\succeq j\}| + |f(V_1\{\succeq j\})| \\ &\leq |U\{\succeq j\}| + |V_2\{\succeq j\}| \leq |(U \cup V_2)\{\succeq j\}|\end{aligned}$$

for all j. □

Proof of Lemma 9.43. Note first that by Proposition 9.40, for every $[B]\langle W\rangle \in \mathbb{S}_B$ with $b = \mathrm{bot}(B)$ it holds that $U \cap G_b^{\triangledown} = (U_H \dot\cup U_B) \cap G_b^{\triangledown}$ blocks $[B]\langle W\rangle$. Therefore, all vertices in U_B needed to block $[B]\langle W\rangle$ can be found in $U_B \cap G_b^{\triangledown}$. Rephrasing this slightly, the blob-pebbling configuration $\mathbb{S}$ is blocked by $U_H \dot\cup \bigl(U_B \cap \bigcup_{b\in\mathcal{B}_B} G_b^{\triangledown}\bigr)$, and since U is subset-minimal we get that

$$U_B = U_B \cap \textstyle\bigcup_{b\in\mathcal{B}_B} G_b^{\triangledown} \ . \tag{9.61}$$

Consider the bipartite graph with $\mathcal{B}_B$ and U_B as the left- and right-hand vertices, where the neighbours of each $b \in \mathcal{B}_B$ are the vertices $N(b) = U_B \cap G_b^{\triangledown}$ in U_B above b. We have that $N(\mathcal{B}_B) = U_B \cap \bigcup_{b\in\mathcal{B}_B} G_b^{\triangledown} = U_B$ by (9.61). Let $\mathcal{B}' \subseteq \mathcal{B}_B$ be a largest set such that $\bigl|N(\mathcal{B}')\bigr| < \bigl|\mathcal{B}'\bigr|$. If $\mathcal{B}' = \mathcal{B}_B$ we are done since this is the inequality $|U_B| < |\mathcal{B}_B|$. Suppose therefore that $\mathcal{B}' \subsetneqq \mathcal{B}_B$ and $|U_B| = |N(\mathcal{B}_B)| > |\mathcal{B}_B|$.

For all $\mathcal{B}'' \subseteq \mathcal{B}_B \setminus \mathcal{B}'$ we must have $\bigl|N(\mathcal{B}'') \setminus N(\mathcal{B}')\bigr| \geq \bigl|\mathcal{B}''\bigr|$, for otherwise $\mathcal{B}''$ could be added to $\mathcal{B}'$ to yield an even larger set $\mathcal{B}^* = \mathcal{B}' \cup \mathcal{B}''$ with $\bigl|N(\mathcal{B}^*)\bigr| < |\mathcal{B}^*|$ contrary to the assumption that $\mathcal{B}'$ has maximal size among all sets with this property. It follows by Hall's marriage theorem that there must exist a matching of $\mathcal{B}_B \setminus \mathcal{B}'$ into $N(\mathcal{B}_B \setminus \mathcal{B}') \setminus N(\mathcal{B}') = U_B \setminus N(\mathcal{B}')$. Thus, $\bigl|\mathcal{B}_B \setminus \mathcal{B}'\bigr| \leq \bigl|U_B \setminus N(\mathcal{B}')\bigr|$ and in addition it follows from the way our bipartite graph is constructed that every $b \in \mathcal{B}_B \setminus \mathcal{B}'$ is matched to some $u \in U_B \setminus N(\mathcal{B}')$ with $\mathrm{level}(u) \geq \mathrm{level}(b)$.

Clearly, all subconfigurations in

$$\mathbb{S}_B^1 = \bigl\{[B]\langle W\rangle \in \mathbb{S}_B \bigm| \mathrm{bot}(B) \in \mathcal{B}_B \setminus \mathcal{B}'\bigr\} \tag{9.62}$$

are blocked by $\mathcal{B}_B \setminus \mathcal{B}'$ (even hidden by this set, to be precise). Also, as was argued in the beginning of the proof, every $[B]\langle W\rangle \in \mathbb{S}_B$ with $b = \mathrm{bot}(B)$ is blocked by $U_H \cup \bigl(U_B \cap G_b^{\triangledown}\bigr) = U_H \cup N(b)$, so all subconfigurations in

$$\mathbb{S}_B^2 = \bigl\{[B]\langle W\rangle \in \mathbb{S}_B \bigm| \mathrm{bot}(B) \in \mathcal{B}'\bigr\} \tag{9.63}$$

are blocked by $U_H \cup N(\mathcal{B}')$ where $\bigl|N(\mathcal{B}')\bigr| < \bigl|\mathcal{B}'\bigr|$. And we know that $\mathbb{S}_H$ is blocked (even hidden) by U_H. It follows that if we let

$$U^* = U_H \cup N(\mathcal{B}') \cup (\mathcal{B}_B \setminus \mathcal{B}') \tag{9.64}$$

we get a vertex set U^* that blocks $\mathbb{S}_H \cup \mathbb{S}_B^1 \cup \mathbb{S}_B^2 = \mathbb{S}$, has measure $m(U^*) \leq m(U)$ because of Observation 9.45, and has size

$$\bigl|U^*\bigr| \leq |U_H| + \bigl|N(\mathcal{B}')\bigr| + \bigl|\mathcal{B}_B \setminus \mathcal{B}'\bigr| < |U_H| + \bigl|\mathcal{B}'\bigr| + \bigl|\mathcal{B}_B \setminus \mathcal{B}'\bigr| = |U| \tag{9.65}$$

strictly less than the size of U. But this is a contradiction, since U was chosen to be of minimal size. The lemma follows. □

The idea in the remaining part of the proof is as follows: Fix some smallest subset $U_H \subseteq U$ that hides $\mathbb{S}_H$, and let $U_B = U \setminus U_H$. Corollary 9.44 says that U_H is the totally dominating part of U and hence that U_H is very large. But U_H hides the blob subconfigurations in $\mathbb{S}_H$ very much in a similar way as for hiding sets in the standard black-white pebble game. And we know from Section 6.5 that such sets need not be very large. Therefore we want to use Klawe-like ideas to derive a contradiction by transforming U_H locally into a (much) better blocking set for $\mathbb{S}_H$. The problem is that this might leave some subconfigurations in $\mathbb{S}_B$ not being blocked any longer (note that in general U_B will *not* on its own block $\mathbb{S}_B$). However, since we have chosen our parameter $C_K = 13$ for the Generalized LHC property 9.23 so generously and since the transformation in Section 6.5 works for the (non-generalized) LHC property with parameter 1, we expect our locally transformed blocking set to be so much cheaper that we can afford to take care of any subconfigurations in $\mathbb{S}_B$ that are no longer blocked simply by adding all bottom vertices for all black blobs in these subconfigurations to the blocking set.

We will not be able to pull this off by just making one local improvement of the hiding set as was done in Section 6.5, though. The reason is that the local improvement to U_H could potentially be very small, but lead to very many subconfigurations in $\mathbb{S}_B$ becoming unblocked. If so, we cannot afford adding new vertices blocking these subconfigurations without risking to increase the size and/or potential of our new blocking set too much. To make sure that this does not happen, we instead make multiple local improvements of U_H simultaneously. Our next lemma says that we can do this without losing control of how the measure behaves.

Lemma 9.46 (Generalization of Lemma 6.40). *Suppose $U_1, \ldots, U_k, V_1, \ldots, V_k$, and Y are vertex sets in a layered graph such that for all $i, j \in [k]$ with $i \neq j$ it holds that $U_i \precsim_m V_i$, $V_i \cap V_j = \emptyset$, $U_i \cap V_j = \emptyset$ and $Y \cap V_i = \emptyset$. Then $m\bigl(Y \cup \bigcup_{i=1}^{k} U_i\bigr) \leq m\bigl(Y \cup \bigcup_{i=1}^{k} V_i\bigr)$.*

Proof. By induction over k. The base case $k = 1$ is Lemma 6.40 on page 78.

For the induction step, let $Y' = Y \cup \bigcup_{i=1}^{k-1} U_i$. Since $U_k \precsim_m V_k$ and $Y' \cap V_k = \emptyset$ by assumption, we get from Lemma 6.40 that

$$m\bigl(Y \cup \textstyle\bigcup_{i=1}^{k} U_i\bigr) = m\bigl(Y' \cup U_k\bigr) \leq \\ m\bigl(Y' \cup V_k\bigr) = m\bigl(Y \cup \textstyle\bigcup_{i=1}^{k-1} U_i \cup V_k\bigr) \ . \quad (9.66)$$

Letting $Y'' = Y \cup V_k$, we see that (again by assumption) it holds for all $i, j \in [k-1]$, $i \neq j$, that $U_i \precsim_m V_i$, $V_i \cap V_j = \emptyset$, $U_i \cap V_j = \emptyset$ and $Y'' \cap V_i = \emptyset$. Hence, by the induction hypothesis we have

$$m\bigl(Y \cup \textstyle\bigcup_{i=1}^{k-1} U_i \cup V_k\bigr) = m\bigl(Y'' \cup \textstyle\bigcup_{k=1}^{i-1} U_i\bigr) \leq \\ m\bigl(Y'' \cup \textstyle\bigcup_{k=1}^{i-1} V_i\bigr) = m\bigl(Y \cup \textstyle\bigcup_{i=1}^{k} V_i\bigr) \quad (9.67)$$

and the lemma follows. □

We also need an observation about the white pebbles in $\mathbb{S}_H$.

Observation 9.47. *For any $[B]\langle W\rangle \in \mathbb{S}_H$ with $b = \mathrm{bot}(B)$ it holds that $W = W \cap G_{\vartriangle}^{b}$.*

Proof. This is so since $\mathbb{S}$ is $\mathcal{W}$-eliminated with respect to U. Since $U \cup W$ hides $b = \mathrm{bot}(B)$, any vertices in $W \cap G_b^{\triangledown}$ are superfluous and will be removed by the $\mathcal{W}$-elimination procedure in Definition 9.31. □

Recalling from (9.56) that $\mathcal{W}_H^{\vartriangle} = \bigl\{W \cap G_{\vartriangle}^{b} \bigm| [B]\langle W\rangle \in \mathbb{S}_H, b = \mathrm{bot}(B)\bigr\}$ this leads to the next, simple but crucial observation.

Observation 9.48. *The vertex set $U_H \cup \mathcal{W}_H^{\vartriangle}$ hides the vertices in $\mathcal{B}_H$ in the sense of Definition 6.16.*

That is, we can consider $\bigl(\mathcal{B}_H, \mathcal{W}_H^{\vartriangle}\bigr)$ to be almost[2] a standard black-white pebble configuration. This sets the stage for applying the machinery of Section 6.5.

Appealing to Lemma 6.28 on page 72, let $X \subseteq U_H \,\dot\cup\, \mathcal{W}_H^{\vartriangle}$ be the unique, minimal tight set such that

$$[\![X]\!] = [\![U_H \,\dot\cup\, \mathcal{W}_H^{\vartriangle}]\!] \tag{9.68}$$

and define

$$\mathcal{W}_T^{\vartriangle} = \mathcal{W}_H^{\vartriangle} \cap X \tag{9.69a}$$

$$U_T = U_H \cap X \tag{9.69b}$$

to be the vertices in $\mathcal{W}_H^{\vartriangle}$ and U_H that remains in X after the bottom-up pruning procedure of Lemma 6.28.

Let $\mathcal{H} = \mathcal{H}(G, X)$ be the hiding set graph of Definition 6.30 for $X = U_T \,\dot\cup\, \mathcal{W}_T^{\vartriangle}$. Suppose that $V_1, \ldots, V_k$ are the connected components of $\mathcal{H}$, and define for $i = 1, \ldots, k$ the vertex sets

$$\mathcal{B}_H^i = \mathcal{B}_H \cap V_i \tag{9.70a}$$

$$\mathcal{W}_H^i = \mathcal{W}_H^{\vartriangle} \cap V_i \tag{9.70b}$$

$$U_H^i = U_H \cap V_i \tag{9.70c}$$

[2] Not quite, since we might have $\mathcal{B}_H \cap \mathcal{W}_H^{\vartriangle} \neq \emptyset$. But at least we know that $U_H \cap \mathcal{W}_H^{\vartriangle} = \emptyset$ by $\mathcal{W}$-elimination and the roles of U and W in $U \cup W$ are fairly indistinguishable in Klawe's proof anyway, so this does not matter.

to be the black, white and "hiding" vertices within component V_i, and

$$\mathcal{W}^i_T = \mathcal{W}^{\triangle}_T \cap V_i \tag{9.70d}$$

$$U^i_T = U_T \cap V_i \tag{9.70e}$$

to be the vertices of $\mathcal{W}^{\triangle}_H$ and U_H in component V_i that "survived" when moving to the tight subset X. Note that we have the disjoint union equalities $\mathcal{W}^{\triangle}_H = \dot{\bigcup}_{i=1}^{k} \mathcal{W}^i_H$, $U_H = \dot{\bigcup}_{i=1}^{k} U^i_H$, et cetera for all of these sets.

Let us also generalize Definition 6.18 of measure and partial measure to multisets of vertices in the natural way, where we charge separately for each copy of every vertex. This is our way of doing the bookkeeping for the extra vertices that might be needed later to block $\mathbb{S}_B$ in the final step of our construction.

This brings us to the key lemma stating how we will locally improve the blocking sets.

Lemma 9.49 (Generalization of Lemma 6.46). *With the assumptions on the blob-pebbling configuration $\mathbb{S}$ and the vertex set U as in Lemma 9.42 and with notation as above, suppose that $U^i_H \cup \mathcal{W}^i_H$ hides $\mathcal{B}^i_H$, that $\mathcal{H}\bigl(U^i_T \cup \mathcal{W}^i_T\bigr)$ is a connected graph, and that*

$$\left|U^i_H\right| \geq 6 \cdot \left|\mathcal{B}^i_H \cup \mathcal{W}^i_H\right| \ . \tag{9.71}$$

Then we can find a multi-set $U^i_* \subseteq [\![U^i_T \cup \mathcal{W}^i_T]\!]$ that hides the vertices in $\mathcal{B}^i_H$, has $\bigl\lfloor |U^i_H|/3 \bigr\rfloor$ extra copies of some fixed but arbitrary vertex on level $L_U = \mathrm{maxlevel}\bigl(U^i_H\bigr)$, and satisfies $U^i_* \precsim_m U^i_H$ and $\left|U^i_*\right| < \left|U^i_H\right|$ (where U^i_* is measured and counted as a multi-set with repetitions).

Proof. Let U^i_* be the set found in Lemma 6.43 on page 79, which certainly is in $[\![U^i_T \cup \mathcal{W}^i_T]\!]$, together with the prescribed extra copies of some (fixed but arbitrary) vertex that we place on level $\mathrm{maxlevel}\bigl([\![U^i_H \cup \mathcal{W}^i_H]\!]\bigr) \geq L_U$ to be on the safe side. By Lemma 6.43, U^i_* hides $\mathcal{B}^i_H$, and the size of U^i_* *counted as a multi-set with repetitions* is

$$\left|U^i_*\right| \leq \left|\mathcal{B}^i_H\right| + \bigl\lfloor |U^i_H|/3 \bigr\rfloor \leq \left(\tfrac{1}{6} + \tfrac{1}{3}\right) \cdot \left|U^i_H\right| < \left|U^i_H\right| \ . \tag{9.72}$$

It remains to show that $U^i_* \precsim_m U^i_H$.

The proof of this last measure inequality is very much as in Lemma 6.46, but with the distinction that the connected graph that we are dealing with is defined over $U^i_T \mathbin{\dot{\cup}} \mathcal{W}^i_T$, but we count the vertices in $U^i_H \mathbin{\dot{\cup}} \mathcal{W}^i_H$. Note, however, that by construction these two unions hide exactly the same set of vertices, i.e.,

$$[\![U^i_T \mathbin{\dot{\cup}} \mathcal{W}^i_T]\!] = [\![U^i_H \mathbin{\dot{\cup}} \mathcal{W}^i_H]\!] \ . \tag{9.73}$$

Recall that by Definition 6.39 on page 78, what we need to do in order to show that $U^i_* \precsim_m U^i_H$ is to find for each j an $l \leq j$ such that $m^j\bigl(U^i_*\bigr) \leq m^l\bigl(U^i_H\bigr)$. As in Lemma 6.46, we divide the proof into two cases.

1. If $j \leq \mathrm{minlevel}\bigl(U^i_T \cup \mathcal{W}^i_T\bigr) = \mathrm{minlevel}\bigl(U^i_H \cup \mathcal{W}^i_H\bigr)$, we get

$$\begin{aligned}
m^j\bigl(U^i_*\bigr) &= j + 2 \cdot \bigl|U^i_*\{\succeq j\}\bigr| && \bigl[\text{ by definition of } m^j(\cdot) \text{ }\bigr] \\
&\leq j + 2 \cdot \bigl|U^i_*\bigr| && \bigl[\text{ since } V\{\succeq j\} \subseteq V \text{ for any } V \text{ }\bigr] \\
&\leq j + 2 \cdot \bigl(|\mathcal{B}^i_H| + \bigl\lfloor |U^i_H|/3 \bigr\rfloor\bigr) && \bigl[\text{ by Lemma 6.43 + extra vertices }\bigr] \\
&< j + 2 \cdot \bigl|U^i_H\bigr| && \bigl[\text{ by the assumption in (9.71) }\bigr] \\
&= j + 2 \cdot \bigl|U^i_H\{\succeq j\}\bigr| && \bigl[\text{ } U^i_H\{\succeq j\} = U^i_H \text{ }\bigr] \\
&= m^j(U^i_H) && \bigl[\text{ by definition of } m^j(\cdot) \text{ }\bigr]
\end{aligned}$$

and we can choose $l = j$ in Definition 6.39.

2. Consider instead $j > \mathrm{minlevel}\bigl(U^i_T \cup \mathcal{W}^i_T\bigr)$ and let $L = \mathrm{minlevel}\bigl(U^i_T \cup \mathcal{W}^i_T\bigr)$. Since the black pebbles in $\mathcal{B}^i_H$ are hidden by $U^i_T \cup \mathcal{W}^i_T$, i.e., $\mathcal{B}^i_H \subseteq \lceil\!\lceil U^i_T \cup \mathcal{W}^i_T \rceil\!\rceil$ in formal notation, recollecting Definition 6.41 and Observation 6.42, part 2, we see that

$$L_{\succeq j}\bigl(\mathcal{B}^i_H\bigr) \leq L_{\succeq j}\bigl(\lceil\!\lceil U^i_T \cup \mathcal{W}^i_T \rceil\!\rceil\bigr) \tag{9.74}$$

for all j. Also, since $U^i_T \cup \mathcal{W}^i_T$ is a hiding-connected vertex set in a spreading graph G, combining Definition 6.44 with the fact that $U^i_T \cup \mathcal{W}^i_T \subseteq U^i_H \cup \mathcal{W}^i_H$ we can derive that

$$j + L_{\succeq j}\bigl(\lceil\!\lceil U^i_T \cup \mathcal{W}^i_T \rceil\!\rceil\bigr) \leq L + \bigl|U^i_T \cup \mathcal{W}^i_T\bigr| \leq L + \bigl|U^i_H \cup \mathcal{W}^i_H\bigr| \ . \tag{9.75}$$

Together, (9.74) and (9.75) say that

$$j + L_{\succeq j}\bigl(\mathcal{B}^i_H\bigr) \leq L + \bigl|U^i_H \cup \mathcal{W}^i_H\bigr| \tag{9.76}$$

and using this inequality we can show that

$$\begin{aligned}
m^j(U^i_*) &= j + 2 \cdot \bigl|U^i_*\{\succeq j\}\bigr| && \bigl[\text{ by definition of } m^j(\cdot) \text{ }\bigr] \\
&\leq j + L_{\succeq j}\bigl(\mathcal{B}^i_H\bigr) + \bigl|\mathcal{B}^i_H\bigr| + 2 \cdot \bigl\lfloor |U^i_H|/3 \bigr\rfloor && \bigl[\text{ by Lemma 6.43 }\bigr] \\
&\leq L + \bigl|U^i_H \cup \mathcal{W}^i_H\bigr| + \bigl|\mathcal{B}^i_H\bigr| + 2 \cdot \bigl\lfloor |U^i_H|/3 \bigr\rfloor && \bigl[\text{ by (9.76) }\bigr] \\
&\leq L + \tfrac{5}{3}\bigl|U^i_H\bigr| + \bigl|\mathcal{B}^i_H\bigr| + \bigl|\mathcal{W}^i_H\bigr| && \bigl[\text{ } |A \cup B| \leq |A| + |B| \text{ }\bigr] \\
&\leq L + \tfrac{5}{3}\bigl|U^i_H\bigr| + 2 \cdot \bigl|\mathcal{B}^i_H \cup \mathcal{W}^i_H\bigr| && \bigl[\text{ } |A| + |B| \leq 2 \cdot |A \cup B| \text{ }\bigr] \\
&\leq L + 2 \cdot \bigl|U^i_H\bigr| && \bigl[\text{ by (9.71) }\bigr] \\
&= L + 2 \cdot |U^i_H\{\succeq L\}| && \bigl[\text{ } L \leq \mathrm{minlevel}(U^i_H) \text{ }\bigr] \\
&= m^L(U^i_H) && \bigl[\text{ by definition of } m^L(\cdot) \text{ }\bigr]
\end{aligned}$$

Thus, the partial measure of U^i_H at the minimum level L is always at least as large as the partial measure of U^i_* at levels j above this minimum level, and we can choose $l = L$ in Definition 6.39.

Consequently, $U^i_* \precsim_m U^i_H$ and the lemma follows. □

Now we want to determine in which connected components of the hiding set graph $\mathcal{H}$ we should apply Lemma 9.49. Loosely put, we want to be sure that changing U^i_H to U^i_* is worthwhile, i.e., that we gain enough from this transformation to compensate for the extra hassle of reblocking blobs in $\mathbb{S}_B$ that turn unblocked when we change U^i_H. With this in mind, let us define the *weight* of a component V_i in $\mathcal{H}$ as

$$w(V_i) = \begin{cases} \lceil |U^i_H|/6 \rceil & \text{if } \left|U^i_H\right| \geq 6 \cdot \left|\mathcal{B}^i_H \cup \mathcal{W}^i_H\right|, \\ 0 & \text{otherwise.} \end{cases} \tag{9.77}$$

The idea is that a component V_i has large weight if the hiding set U^i_H in this component is large compared to the number of bottom black vertices in $\mathcal{B}^i_H$ hidden and the white pebbles $\mathcal{W}^i_H$ helping U^i_H to hide $\mathcal{B}^i_H$. If we concentrate on changing the hiding sets in components with non-zero weight, we hope to gain more from the transformation of U^i_H into U^i_* than we lose from then having to reblocking $\mathbb{S}_B$. And since U_H is large, the total weight of the non-zero-weight components is guaranteed to be reasonably large.

Proposition 9.50. *With notation as above, the total weight of all connected components $V_1, \ldots, V_k$ in the hiding set graph $\mathcal{H} = \mathcal{H}\big(G, U_T \cup \mathcal{W}^{\vartriangle}_T\big)$ is $\sum_{i=1}^k w(V_i) > \left|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}^{\vartriangle}_H\right|$.*

Proof. The total size of the union of all subsets $U^i_H \subseteq U_H$ with sizes $\left|U^i_H\right| < 6 \cdot \left|\mathcal{B}^i_H \cup \mathcal{W}^i_H\right|$ resulting in zero-weight components V_i in $\mathcal{H}$ is clearly strictly less than

$$6 \cdot \sum_{i=1}^{k} \left|\mathcal{B}^i_H \cup \mathcal{W}^i_H\right| = 6 \cdot \left|\mathcal{B}_H \cup \mathcal{W}^{\vartriangle}_H\right| \leq 6 \cdot \left|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}^{\vartriangle}_H\right| \ . \tag{9.78}$$

Since according to Corollary 9.44 we have that $\left|U_H\right| \geq 12 \cdot \left|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}^{\vartriangle}_H\right|$, it follows that the size of the union $\bigcup_{w(V_i)>0} U^i_H$ of all subsets U^i_H corresponding to non-zero-weight components V_i must be strictly larger than $6 \cdot \left|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}^{\vartriangle}_H\right|$. But then

$$\sum_{w(V_i)>0} w(V_i) \geq \sum_{w(V_i)>0} \lceil |U^i_H|/6 \rceil \geq \frac{1}{6} \cdot \left| \bigcup_{w(V_i)>0} U^i_H \right| > \left|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}^{\vartriangle}_H\right| \tag{9.79}$$

as claimed in the proposition. □

We have now collected all tools needed to establish the Generalized limited hiding-cardinality property for spreading graphs. Before we wrap up the proof, let us recapitulate what we have shown so far.

We have divided the blocking set U into a disjoint union $U_H \dot{\cup} U_B$ of the vertices U_H not only blocking but actually *hiding* the subconfigurations in $\mathbb{S}_H \subseteq \mathbb{S}$, and the vertices U_B just helping U_H to block the remaining subconfigurations in $\mathbb{S}_B = \mathbb{S} \backslash \mathbb{S}_H$.

In Lemma 9.43 and Corollary 9.44, we proved that if U is large (which we are assuming) then U_B must be very small compared to U_H, so we can basically just ignore U_B. If we want to do something interesting, it will have to be done with U_H.

And indeed, Lemma 9.49 tells us that we can restructure U_H to get a new vertex set hiding $\mathbb{S}_H$ and make considerable savings, but that this can lead to $\mathbb{S}_B$ no longer being blocked. By Proposition 9.50, there is a large fraction of U_H that resides in the non-zero-weight components of the hiding set graph $\mathcal{H}$ (as defined in Equation (9.77)). We would like to show that by judiciously performing the restructuring of Lemma 9.49 in these components, we can also take care of $\mathbb{S}_B$.

More precisely, we claim that we can combine the hiding sets U^i_* obtained in Lemma 9.49 with some subsets of $U_H \cup U_B$ and $\mathcal{B}_B$ into a new blocking set U^* for all of $\mathbb{S}_H \cup \mathbb{S}_B = \mathbb{S}$ in such a way that the measure $m\bigl(U^*\bigr)$ does not exceed $m(U) = \mathrm{pot}(\mathbb{S})$ but so that $\bigl|U^*\bigr| < |U|$. But this contradicts the assumptions in Lemma 9.42. It follows that the conclusion in Lemma 9.42, which we assumed to be false in order to derive a contradiction, must instead be true. That is, any set U that is chosen as in Lemma 9.42 must have size $|U| \leq 13 \cdot \bigl|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}^{\triangle}_H\bigr|$. This in turn implies Theorem 9.35, i.e., that layered spreading graphs possess the Generalized limited hiding-cardinality property that we assumed in order to get a lower bound on blob-pebbling price, and we are done.

We proceed to establish this final claim. Our plan is once again to do some bipartite matching with the help of Hall's theorem. Create a weighted bipartite graph with the vertices in $\mathcal{B}_B = \bigl\{\mathrm{bot}(B)\bigm|[B]\langle W\rangle \in \mathbb{S}_B\bigr\}$ on the left-hand side and with the non-zero-weight connected components among $V_1, \ldots, V_k$ in $\mathcal{H}$ in the sense of (9.77) acting as "supervertices" on the right-hand side. Reorder the indices among the connected components $V_1, \ldots, V_k$ if needed so that the non-zero-weight components are $V_1, \ldots, V_{k'}$. All vertices in the weighted graphs are assigned weights so that each right-hand side supervertex V_i gets its weight according to (9.77), and each left-hand vertex has weight 1.[3] We define the neighbours of each fixed vertex $b \in \mathcal{B}_B$ to be

$$N(b) = \bigl\{V_i \bigm| w(V_i) > 0 \text{ and } \mathrm{maxlevel}\bigl(U^i_H\bigr) > \mathrm{level}(b)\bigr\} \ , \tag{9.80}$$

i.e., all non-zero-weight components V_i that contain vertices in the hiding set U_H that could possibly be involved in blocking any subconfiguration $[B]\langle W\rangle \in \mathbb{S}_B$ having bottom vertex $\mathrm{bot}(B) = b$. This is so since by Proposition 9.40, any vertex $u \in U_H$ helping to block such a subconfiguration $[B]\langle W\rangle \in \mathbb{S}_B$ must be strictly above b. Thus, if the highest-level vertices in U^i_H are on a level below b, no vertex in U^i_H can be responsible for blocking $[B]\langle W\rangle$.

Let $\mathcal{B}' \subseteq \mathcal{B}_B$ be a largest set such that $w\bigl(N\bigl(\mathcal{B}'\bigr)\bigr) \leq \bigl|\mathcal{B}'\bigr|$. We must have

$$N\bigl(\mathcal{B}'\bigr) \neq \textstyle\bigcup_{i=1}^{k'} V_i \tag{9.81}$$

[3] Or, if we like, we can equivalently think of an unweighted graph, where each V_i is a cloud of $w(V_i)$ unique and distinct vertices, and where $N(b)$ in (9.80) always contains either all or none of these vertices.

since $w\bigl(\bigcup_{i=1}^{k'} V_i\bigr) > \bigl|\mathcal{B}_H \cup \mathcal{B}_B \cup \mathcal{W}_H^{\triangle}\bigr| \geq \bigl|\mathcal{B}_B\bigr|$ by Proposition 9.50. For all $\mathcal{B}'' \subseteq \mathcal{B}_B \setminus \mathcal{B}'$ it holds that

$$w\bigl(N(\mathcal{B}'') \setminus N(\mathcal{B}')\bigr) \geq \bigl|\mathcal{B}''\bigr| \tag{9.82}$$

since otherwise $\mathcal{B}'$ would not be of largest size as assumed above. The inequality (9.82) plugged into Hall's marriage theorem tells us that there is a matching of the vertices in $\mathcal{B}_B \setminus \mathcal{B}'$ to the components in $\bigcup_{i=1}^{k'} V_i \setminus N\bigl(\mathcal{B}'\bigr) \neq \emptyset$ with the property that no component V_i gets matched with more than $w(V_i)$ vertices from $\mathcal{B}_B \setminus \mathcal{B}'$.

Reorder the components in the hiding set graph $\mathcal{H}$ so that the matched components in $\mathcal{H}$ are $V_1, \ldots, V_m$ and the rest of the components are $V_{m+1}, \ldots, V_k$ and so that $U_H^1, \ldots, U_H^m$ and $U_H^{m+1}, \ldots, U_H^k$ are the corresponding subsets of the hiding set U_H. Then pick good local blockers $U_*^i \subseteq V_i$ as in Lemma 9.49 for all components $V_1, \ldots, V_m$. Now the following holds:

1. By construction and assumption, respectively, we know that the vertex set union $\bigcup_{i=1}^{m} U_*^i \cup \bigcup_{i=m+1}^{k} U_H^i$ blocks (and even hides) $\mathbb{S}_H$.

2. All subconfigurations in

$$\mathbb{S}_B^1 = \bigl\{[B]\langle W\rangle \in \mathbb{S}_B \bigm| \mathrm{bot}(B) \in \mathcal{B}'\bigr\} \tag{9.83}$$

   are blocked by $U_B \cup N\bigl(\mathcal{B}'\bigr) = U_B \cup \bigcup_{i=m+1}^{k} U_H^i$, as we have not moved any elements in U above $\mathcal{B}'$.

3. With notation as in Lemma 9.46, let $Y = U_B \cup \bigcup_{i=m+1}^{k} U_H^i$ and consider U_*^i and U_H^i for $i = 1, \ldots, m$. We have $U_*^i \precsim_m U_H^i$ for $i = 1, \ldots, m$ by Lemma 9.49. Also, since $U_H \cap U_B = \emptyset$ and $U_*^i \subseteq V_i$ and $U_H^i \subseteq V_i$ for $V_1, \ldots, V_k$ pairwise disjoint sets of vertices, it holds for all $i, j \in [m]$, $i \neq j$, that $U_*^i \cap U_*^j = \emptyset$, $U_H^i \cap U_H^j = \emptyset$, $U_*^i \cap U_H^j = \emptyset$ and $Y \cap U_H^j = \emptyset$. Therefore, the conditions in Lemma 9.46 are satisfied and we conclude that

$$\begin{aligned} m\bigl(U_B \cup \textstyle\bigcup_{i=1}^{m} U_*^i \cup \bigcup_{i=m+1}^{k} U_H^i\bigr) &= m\bigl(Y \cup \textstyle\bigcup_{i=1}^{m} U_*^i\bigr) \\ &\leq m\bigl(Y \cup \textstyle\bigcup_{i=1}^{m} U_H^i\bigr) \\ &= m\bigl(U_B \cup \textstyle\bigcup_{i=1}^{m} U_H^i \cup \bigcup_{i=m+1}^{k} U_H^i\bigr) \\ &= m(U) \enspace , \end{aligned} \tag{9.84}$$

   where we note that $U_B \cup \bigcup_{i=1}^{m} U_*^i \cup \bigcup_{i=m+1}^{k} U_H^i$ is *measured as a multi-set with repetitions.* Also, we have the strict inequality

$$\bigl|U_B \cup \textstyle\bigcup_{i=1}^{m} U_*^i \cup \bigcup_{i=m+1}^{k} U_H^i\bigr| < |U| \enspace , \tag{9.85}$$

   where again the multi-set is *counted with repetitions.*

4. It remains to take care of the potentially unblocked subconfigurations in

$$\mathbb{S}_B^2 = \bigl\{[B]\langle W\rangle \in \mathbb{S}_B \bigm| \mathrm{bot}(B) \in \mathcal{B}_B \setminus \mathcal{B}'\bigr\} \ . \tag{9.86}$$

But we derived above that there is a matching of $\mathcal{B}_B \setminus \mathcal{B}'$ to $V_1, \ldots, V_m$ such that no V_i is chosen by more than

$$w(V_i) = \bigl\lceil |U_H^i|/6 \bigr\rceil \leq \bigl\lfloor |U_H^i|/3 \bigr\rfloor \tag{9.87}$$

vertices from $\mathcal{B}_B \setminus \mathcal{B}'$ (where we used that $\bigl|U_H^i\bigr| \geq 6$ if $w(V_i) > 0$ to get the last inequality). This means that there is a spare blocker vertex in U_*^i for each $b \in \mathcal{B}_B \setminus \mathcal{B}'$ that is matched to V_i. Also, by the definition of neighbours in our weighted bipartite graph, each b is matched to a component with $\mathrm{maxlevel}\bigl(U_H^i\bigr) > \mathrm{level}(b)$. By Observation 9.45, lowering these spare vertices from $\mathrm{maxlevel}\bigl(U_H^i\bigr)$ to $\mathrm{level}(b)$ can only decrease the measure.

Finally, throw away any remaining multiple copies in our new blocking set, and denote the resulting set by U^*. We have that U^* blocks $\mathbb{S}$ and that $m\bigl(U^*\bigr) \leq m(U)$ but $\bigl|U^*\bigr| < |U|$. This is a contradiction since U was chosen to be of minimal size, and thus Lemma 9.42 must hold. But then Theorem 9.35 follows immediately as well, as was noted above.

9.5.5 Proof Recap for Theorem 2.3 and Optimality of Result

Let us conclude this chapter by recalling why the tight bound on clause space for refuting pebbling contradictions in Theorem 2.3 now follows and by showing that the current construction cannot be pushed to give a better result.

Theorem 9.51 (rephrasing of Theorem 2.3). *Suppose that G_h is a layered blob-pebblable DAG of height h that is spreading. Then the clause space of refuting the pebbling contradiction $Peb_{G_h}^d$ of degree $d > 1$ is $Sp(Peb_{G_h}^d \vdash 0) = \Theta(h)$.*

Proof. The $\mathrm{O}(h)$ upper bound on clause space follows from the bound $\mathsf{Peb}(G_h) \leq h + \mathrm{O}(1)$ on the black pebbling price in Lemma 6.9 on page 61 combined with the bound $Sp(Peb_G^d \vdash 0) \leq \mathsf{Peb}(G) + \mathrm{O}(1)$ from Proposition 5.10 on page 53.

For the lower bound, we instead consider the pebbling formula $^*Peb_{G_h}^d$ without target axioms $\overline{x(z)}_1, \ldots, \overline{x(z)}_d$ and use that by Lemma 8.9 on page 104 it holds that $Sp\bigl(Peb_{G_h}^d \vdash 0\bigr) = Sp\bigl({}^*Peb_{G_h}^d \vdash \bigvee_{i=1}^d x(z)_i\bigr)$. Fix any resolution derivation $\pi : {}^*Peb_{G_h}^d \vdash \bigvee_{i=1}^d x(z)_i$ and let $\mathcal{P}_\pi$ be the complete blob-pebbling of the graph G associated to π in Theorem 9.10 such that $\mathsf{cost}(\mathcal{P}_\pi) \leq \max_{\mathbb{C}\in\pi}\bigl\{\mathsf{cost}(\mathbb{S}(\mathbb{C}))\bigr\} + \mathrm{O}(1)$. On the one hand, Theorem 9.16 says that $\mathsf{cost}(\mathbb{S}(\mathbb{C})) \leq |\mathbb{C}|$ provided that $d > 1$, so in particular it must hold that $\mathsf{cost}(\mathcal{P}_\pi) \leq Sp(\pi) + \mathrm{O}(1)$. On the other hand, $\mathsf{cost}(\mathcal{P}_\pi) \geq \mathsf{Blob\text{-}Peb}(G_h)$ by definition, and by Theorems 9.24 and 9.35 it holds that $\mathsf{Blob\text{-}Peb}(G_h) = \Omega(h)$. Thus $Sp(\pi) = \Omega(h)$, and the theorem follows. □

Plugging in pyramid graphs Π_h in Theorem 9.51, we get k-CNF formulas F_n of size $\Theta(n)$ with refutation clause space $\Theta(\sqrt{n})$. This is the best we can get from pebbling formulas over spreading graphs.

Theorem 9.52. *Let G be any layered spreading graph and suppose that Peb^d_G has formula size and number of clauses $\Theta(n)$. Then $Sp\bigl(Peb^d_G \vdash 0\bigr) = \mathrm{O}(\sqrt{n})$.*

Proof. Suppose that G has height h. Then $Sp\bigl(Peb^d_G \vdash 0\bigr) = \mathrm{O}(h)$ as was noted above. The size of Peb^d_G, as well as the number of clauses, is linear in the number of vertices $|V(G)|$. We claim that the fact that G is spreading implies that $|V(G)| = \Omega\bigl(h^2\bigr)$, from which the theorem follows.

To prove the claim, let V_L denote the vertices of G on level L. Then $|V(G)| = \sum_{L=0}^{h}|V_L|$. Obviously, for any L the set V_L hides the sink z of G. Fix for every L some arbitrary minimal subset $V'_L \subseteq V_L$ hiding z. Then V'_L is tight, the graph $\mathcal{H}(V'_L)$ is hiding-connected by Corollary 6.33, and setting $j = h$ in the spreading inequality (6.10) we get that $\left|V'_L\right| \geq 1 + h - L$. Hence $|V(G)| \geq \sum_{L=0}^{h}|V'_L| = \Omega\bigl(h^2\bigr)$. □

The proof of Theorem 9.52 can also be extended to cover the original definition in [49] of spreading graphs that are not necessarily layered, but we omit the details.

Chapter 10

Short Proofs May Be Spacious

In this chapter, we finally resolve the open question about the relationship between refutation clause space and refutation length in resolution by establishing an optimal separation of these two proof complexity measures. This result is joint work with Eli Ben-Sasson [20].

In the first section of this chapter, we state a special case of our result and give an overview of the proof. In subsequent sections, we present the full proof of a somewhat more general result. Some intuition, terminology and notation is repeated from previous chapters in order to make the exposition self-contained for readers skipping directly to this optimal result.

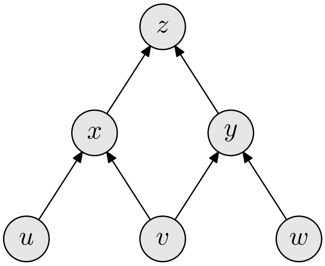
We remark that although the lower bounds on space in this chapter are strictly better than those in Chapters 8 and 9, the results are, in a sense, nevertheless incomparable since we are proving lower bounds for different formulas families.

10.1 Outline of Proof of a Special Case

The key to the optimal separation is to study more general variants of pebbling contradictions. Namely, given a DAG G, we fix some non-constant Boolean function $f_d : \{0,1\}^d \mapsto \{0,1\}$ and define a pebbling contradiction $Peb^d_G[f]$ as the conjunction of clauses encoding the following statements:

- For each source vertex s, $f_d(s_1,\ldots,s_d)$ holds.
- For a non-source vertex r with immediate predecessors p and q, the implication $\bigl(f_d(p_1,\ldots,p_d) \wedge f_d(q_1,\ldots,q_d)\bigr) \rightarrow f_d(r_1,\ldots,r_d)$ holds.
- For the (unique) sink vertex z, $\neg f_d(z_1,\ldots,z_d)$ holds.

Traditionally, the function $f_d(v_1,\ldots,v_d)$ has been $\bigvee_{i=1}^{d} v_i$, and these formulas $Peb^d_G[\vee]$ are also the variant of pebbling contradictions studied in Chapters 8 and 9. In this chapter, we instead use pebbling contradictions $Peb^d_G[\oplus]$ where



(a) Our old friend the pyramid graph Π_2 of height 2.

$$
\begin{array}{ll}
\ (u_1 \lor u_2) & \land\ (v_1 \lor \overline{v}_2 \lor \overline{w}_1 \lor w_2 \lor y_1 \lor y_2) \\
\land\ (\overline{u}_1 \lor \overline{u}_2) & \land\ (v_1 \lor \overline{v}_2 \lor \overline{w}_1 \lor w_2 \lor \overline{y}_1 \lor \overline{y}_2) \\
\land\ (v_1 \lor v_2) & \land\ (\overline{v}_1 \lor v_2 \lor w_1 \lor \overline{w}_2 \lor y_1 \lor y_2) \\
\land\ (\overline{v}_1 \lor \overline{v}_2) & \land\ (\overline{v}_1 \lor v_2 \lor w_1 \lor \overline{w}_2 \lor \overline{y}_1 \lor \overline{y}_2) \\
\land\ (w_1 \lor w_2) & \land\ (\overline{v}_1 \lor v_2 \lor \overline{w}_1 \lor w_2 \lor y_1 \lor y_2) \\
\land\ (\overline{w}_1 \lor \overline{w}_2) & \land\ (\overline{v}_1 \lor v_2 \lor \overline{w}_1 \lor w_2 \lor \overline{y}_1 \lor \overline{y}_2) \\
\land\ (u_1 \lor \overline{u}_2 \lor v_1 \lor \overline{v}_2 \lor x_1 \lor x_2) & \land\ (x_1 \lor \overline{x}_2 \lor y_1 \lor \overline{y}_2 \lor z_1 \lor z_2) \\
\land\ (u_1 \lor \overline{u}_2 \lor v_1 \lor \overline{v}_2 \lor \overline{x}_1 \lor \overline{x}_2) & \land\ (x_1 \lor \overline{x}_2 \lor y_1 \lor \overline{y}_2 \lor \overline{z}_1 \lor \overline{z}_2) \\
\land\ (u_1 \lor \overline{u}_2 \lor \overline{v}_1 \lor v_2 \lor x_1 \lor x_2) & \land\ (x_1 \lor \overline{x}_2 \lor \overline{y}_1 \lor y_2 \lor z_1 \lor z_2) \\
\land\ (u_1 \lor \overline{u}_2 \lor \overline{v}_1 \lor v_2 \lor \overline{x}_1 \lor \overline{x}_2) & \land\ (x_1 \lor \overline{x}_2 \lor \overline{y}_1 \lor y_2 \lor \overline{z}_1 \lor \overline{z}_2) \\
\land\ (\overline{u}_1 \lor u_2 \lor v_1 \lor \overline{v}_2 \lor x_1 \lor x_2) & \land\ (\overline{x}_1 \lor x_2 \lor y_1 \lor \overline{y}_2 \lor z_1 \lor z_2) \\
\land\ (\overline{u}_1 \lor u_2 \lor v_1 \lor \overline{v}_2 \lor \overline{x}_1 \lor \overline{x}_2) & \land\ (\overline{x}_1 \lor x_2 \lor y_1 \lor \overline{y}_2 \lor \overline{z}_1 \lor \overline{z}_2) \\
\land\ (\overline{u}_1 \lor u_2 \lor \overline{v}_1 \lor v_2 \lor x_1 \lor x_2) & \land\ (\overline{x}_1 \lor x_2 \lor \overline{y}_1 \lor y_2 \lor z_1 \lor z_2) \\
\land\ (\overline{u}_1 \lor u_2 \lor \overline{v}_1 \lor v_2 \lor \overline{x}_1 \lor \overline{x}_2) & \land\ (\overline{x}_1 \lor x_2 \lor \overline{y}_1 \lor y_2 \lor \overline{z}_1 \lor \overline{z}_2) \\
\land\ (v_1 \lor \overline{v}_2 \lor w_1 \lor \overline{w}_2 \lor y_1 \lor y_2) & \land\ z_1 \lor \overline{z}_2 \\
\land\ (v_1 \lor \overline{v}_2 \lor w_1 \lor \overline{w}_2 \lor \overline{y}_1 \lor \overline{y}_2) & \land\ \overline{z}_1 \lor z_2
\end{array}
$$

(b) The corresponding XOR-pebbling contradiction.

Figure 10.1: XOR-pebbling contradiction $Peb^2_{\Pi_2}[\oplus]$ for the pyramid graph Π_2.

$f_d(v_1, \ldots, v_d) = \bigoplus_{i=1}^{d} v_i$ denotes the exclusive or of the variables. A small example of such an *XOR-pebbling contradiction* is presented in Figure 10.1. For such formulas, it turns out that we can prove lower bounds on clause space in terms of pebbling price for any DAG G.

Theorem 2.5 (restated). *The space of refuting XOR-pebbling contradictions over any DAG G in resolution is lower-bounded by the black-white pebbling price of G, provided that the number of variables per vertex is at least* 2.

If we fix the number of variables per vertex and study DAGs with constant fan-in, it is easy to show that XOR-pebbling contradictions can be refuted in linear length and constant width. Using the family of DAGs provided by Theorem 5.5 and proving Theorem 2.5 in a slightly more general setting than stated above, we get our optimal separation of space and length.

Corollary 2.6 (restated). *For all $k \geq 6$ there is a family $\{F_n\}_{n=1}^{\infty}$ of k-CNF formulas of size $\mathrm{O}(n)$ that can be refuted in resolution in length $L(F_n \vdash 0) = \mathrm{O}(n)$ and width $W(F_n \vdash 0) = \mathrm{O}(1)$ but require space $Sp(F_n \vdash 0) = \Omega(n/\log n)$.*

As has been discussed previously, a refutation in length $\mathrm{O}(n)$ is always possible to carry out in space $\mathrm{O}(n/\log n)$, so the separation of space and length in Corollary 2.6 is asymptotically optimal. Since any (unsatisfiable) formula of size $\mathrm{O}(n)$ is refutable in width $\mathrm{O}(n)$, the separation of space and width is also very nearly optimal, except for perhaps by a logarithmic factor. As an extra bonus, we note that while the constructions in Chapters 8 and 9 are quite intricate and the proofs very involved, our optimal lower bound proof is relatively clean and straightforward.

10.1.1 Recap of Intuitive Proof Idea

All along, we have been trying to give a proof of a lower bound on the resolution refutation space of pebbling contradictions structured as follows:

1. First, find a natural interpretation of clause configurations in a refutation of the formula $Peb_G^d[f]$ in terms of black and white pebbles on the DAG G.

2. Then, prove that this interpretation captures pebbling in the sense that for any refutation of Peb_G^d, consecutive clause configurations translates into consecutive pebble configurations in a legal black-white pebbling of G.

3. Finally, show that the interpretation captures space in the sense that if a clause set induces N pebbles, then it must contain at least N clauses.

Unfortunately, as we have seen this idea does not quite work "off the shelf." Pebblings of DAGs and resolution refutations of CNF formulas are very different objects, and there is no reason a priori that there should be a tight connection between the two. However, relaxing our requirements for the correspondence, in Chapters 8 and 9 we made essentially the proof idea above work for two special cases. In the rest of this section, we describe the modifications made in Chapters 8 and 9 and then outline how, using our new approach, we can make the bits and pieces fit together to yield the optimal results in Theorem 2.5 and Corollary 2.6.

10.1.2 Formalizing the Idea and Encountering Problems

Recall that the black-white pebble game played on a DAG G can be viewed as a way of proving the end result of the calculation described by G. Black pebbles denote proven partial results of the computation. White pebbles denote assumptions about partial results which have been used to derive other partial results (i.e., black pebbles), but these assumptions will have to be verified for the calculation to be complete. The final goal is a black pebble on the sink z and no other pebbles in the graph, corresponding to an unconditional proof of the end result of the calculation.

Translating this to pebbling contradictions $Peb^d_G[f]$, it turns out that a fruitful way to think of a black pebble on v is that it should correspond to "truth of $f_d(v_1, \ldots, v_d)$". A white pebble on a vertex w can be understood to mean that we need to *assume* the partial result on w to derive the black pebbles above w in the graph. Needing to assume the truth of w is the opposite of knowing the truth of w, so extending the reasoning above we get that a white-pebbled vertex should correspond to "falsity of $f_d(v_1, \ldots, v_d)$".

Developing this intuitive correspondence one step further, we find that it seems natural that a set of clauses $\mathbb{C}$ on the blackboard should be translated into a black pebble on v "supported by" white pebbles on $w \in W$ below v if $\mathbb{C}$ implies that if $f_d(w_1, \ldots, w_d)$ holds for all $w \in W$, then $f_d(v_1, \ldots, v_d)$ must also hold, i.e., if the implication

$$\mathbb{C} \vDash f_d(v_1, \ldots, v_d) \vee \bigvee_{w \in W} \neg f_d(w_1, \ldots, w_d) \tag{10.1}$$

is true. And if $\mathbb{C}$ satisfies many different implications on the form (10.1), then $\mathbb{C}$ should correspond to many such black pebbles with associated white pebbles. The idea to interpret clauses in terms of pebbles as in (10.1) by looking not at the structure of the clauses in $\mathbb{C}$ but only on the "semantic content", i.e., which set of clauses $\mathbb{C}$ implies, was the key insight in Chapter 8 and is also used in Chapter 9 as well as the present chapter.

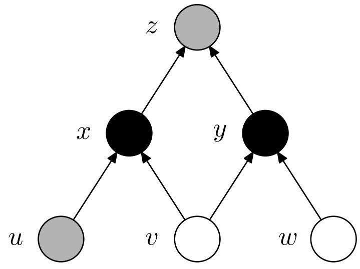
To make this more concrete, Figure 10.2 presents an example of such a translation for a set of clauses derived from the XOR-pebbling contradiction $Peb^2_{\Pi_2}[\oplus]$ in Figure 10.1. It is straightforward to verify that the clause set $\mathbb{C}$ on the blackboard in Figure 10.2(a) implies $x_1 \oplus x_2$ and that $\mathbb{C} \cup \{v_1 \oplus v_2, w_1 \oplus w_2\}$ implies $y_1 \oplus y_2$, so we get black pebbles on x and y and white pebbles on v and w in Figure 10.2(b). This translation from clauses to pebbles is arguably quite straightforward, and it seems that it should yield legal black-white pebblings for all "well-behaved" resolution refutations using some kind of pebbling strategy as the guiding idea when refuting $Peb^d_G[f]$.

The problem is that we have no guarantee that the resolution refutations will be "well-behaved". On the one hand, it might appear plausible that for a formula defined in terms of pebble games, the refutation would somehow need to have a pebbling of G in mind when refuting a pebbling contradiction over G. On the other hand, for all we know there might be some shortcut exploiting other properties of the formula, and maybe the refutation making this shortcut produces clauses that do not fit into the framework sketched above.

Although it is hard to give an example that is at the same time both small and convincing, this turns out to be a serious problem. Some clauses that look strange we can just ignore, but it turns out that a particularly tricky case is when clauses

$$\begin{bmatrix} x_1 \lor x_2 \\ \overline{x}_1 \lor \overline{x}_2 \\ \overline{v}_1 \lor w_1 \lor y_1 \lor y_2 \\ \overline{v}_1 \lor w_1 \lor \overline{y}_1 \lor \overline{y}_2 \\ v_2 \lor w_1 \lor y_1 \lor y_2 \\ v_2 \lor w_1 \lor \overline{y}_1 \lor \overline{y}_2 \\ \overline{v}_1 \lor \overline{w}_2 \lor y_1 \lor y_2 \\ \overline{v}_1 \lor \overline{w}_2 \lor \overline{y}_1 \lor \overline{y}_2 \\ v_2 \lor \overline{w}_2 \lor y_1 \lor y_2 \\ v_2 \lor \overline{w}_2 \lor \overline{y}_1 \lor \overline{y}_2 \end{bmatrix}$$

(a) Clauses on blackboard.



(b) Corresponding pebbles in the graph.

Figure 10.2: Translating XOR-pebbling contradiction clauses to pebbles.

are derived that imply that $f_d(b_1, \ldots, b_d)$ is true not for a single vertex b but for *some* vertex in a set B of size $|B| > 1$, i.e., when we have implications on the form

$$\mathbb{C} \vDash \bigvee_{b \in B} f_d(b_1, \ldots, b_d) \lor \bigvee_{w \in W} \neg f_d(w_1, \ldots, w_d) \ . \tag{10.2}$$

On the one hand it can be shown that we cannot afford to ignore such clauses, but on the other hand there appears to be no way that we can interpret them in terms of black and white pebbles without making some component in the proof idea in Section 10.1.1 break down.

The problem with implications such as (10.2) is something that arises not only in the current chapter but also in Chapters 8 and 9, but the ways to circumvent this problem is very different in the three chapters. In the remaining part of this subsection, we discuss what ideas were explored in Chapters 8 and 9, leading up our current construction.

The key idea in Chapter 8 was to relax the translation function to saying, roughly, that clauses resulting in implications on the form (10.2) are interpreted as a black pebble on the lowermost vertex b of B and white pebbles on all vertices $W' \subseteq W$ below b. This translation was shown to capture clause space in the sense that the number of pebbles are at most the number of clauses. However, in order to accommodate the pebble movements that can result during a refutation, the pebbling rules had to be changed to allow "sliding moves" of white pebbles upwards and black pebbles downwards. For this new game it could be shown that known lower bounds on black-white pebbling break down completely even for very simple graphs, for instance for pyramid graphs. But for the simplest case, namely DAGs that are trees, we established that this new game is essentially equivalent to the standard black-white pebble game, yielding a tight logarithmic bound on clause space for pebbling contradictions over trees.

To cope with more general DAGs, in Chapter 9 the notion that clauses should correspond to black and white *pebbles* was sacrificed. Instead, another pebble game (of sorts) was invented, with white pebbles just as before, but with black *blobs* that can cover multiple vertices instead of single-vertex black pebbles. A blob on a vertex set B can be thought of saying that $f_d(b_1, \ldots, b_d)$ holds for some $b \in B$, and the implication (10.2) is interpreted as corresponding to white pebbles on W and a single black blob on B. When the rules of this blob-pebble game are defined in the right way, it can be shown that resolution refutations of pebbling contradictions over G correspond to pebblings of G for a very general class of DAGs, including (a slight modification of) the DAGs $\{G_n\}_{n=1}^{\infty}$ in Theorem 5.5 of size $\mathrm{O}(n)$ having black-white pebbling price $\Omega(n/\log n)$.

Sadly, now proving lower bounds on pebbling price in our blob-pebble game becomes a formidable challenge. Difficulties arise because we can no longer charge for every vertex covered by a black pebble but only at most 1 for every distinct black blob, and sometimes even very much less than that if the blobs intersect too much. Also, we cannot charge for all white pebbles but only for pebbles located below blobs. Therefore, we are not able to prove lower bounds on blob-pebbling price for general DAGs (and in particular not for the Gilbert-Tarjan DAGs in Theorem 5.5. However, using ideas from [49] proving bounds on black-white pebbling price for pyramid graphs and a class of related DAGs, in Chapter 9 we showed corresponding tight bounds on blob-pebbling price for (almost) the same class of DAGs, resulting, in particular, in a tight $\Theta(\sqrt{n})$ bound on space.

10.1.3 Detailed Outline of Optimal Lower Bound Proof

In this chapter, we, in a sense, unite the two approaches in Chapters 8 and 9. The decisive ingredient in making this work, however, is to look not at standard pebbling contradictions $Peb^d_G[\vee]$ but instead focus on XOR-pebbling contradictions $Peb^d_G[\oplus]$ as in Figure 10.1.

We still cannot hope to get resolution refutations and pebblings to match perfectly, so we devise yet another pebble game, which we call the *resolution-pebbling game*, or just *res-pebbling game* for short. In this game we have ordinary black and white pebbles covering single vertices, but subsets of black pebbles B and white pebbles W are associated in *subconfigurations* $[B]\langle W\rangle$. The pebble configuration at any point in time t consists of a set $\mathbb{R}_t = \bigl\{[B_i]\langle W_i\rangle \bigm| i = 1, \ldots, m\bigr\}$ of such subconfigurations, and the pebbling moves are performed not on individual pebbles but on entire subconfigurations. The pebbling moves are as follows:

Download At any time, derive $[v]\langle pred(v)\rangle$ for $pred(v)$ the immediate predecessors of v.

Resolution From $[B_1]\langle W_1 \cup \{v\}\rangle$ and $[B_2 \cup \{v\}]\langle W_2\rangle$ such that $B_1 \cap W_2 = \emptyset$, derive $[B_1 \cup B_2]\langle W_1 \cup W_2\rangle$.

Weakening From $[B]\langle W\rangle$, derive $[B \cup B']\langle W \cup W'\rangle$ provided that $(B \cup B') \cap (W \cup W') = \emptyset$.

Erasure Any $[B]\langle W\rangle$ may be erased at any time.

The game starts with the graph G being empty, i.e., $\mathbb{R}_0 = \emptyset$, and ends when $\mathbb{R}_\tau = \{[z]\langle\emptyset\rangle\}$ for the unique sink z of G, corresponding to a single black pebble on z and no other pebbles in the DAG.

Let us try to provide some intuition for the pebbling rules. From a pebbling perspective, the download rule is just the familiar fact that we can always place white pebbles on the immediate predecessors of a vertex v and then black-pebble v itself. The resolution rule says roughly that we can remove a white pebble from v if we first place a black pebble there. Since v can only be black-pebbled when the predecessors of v have pebbles on them, this rule is roughly equivalent to the combination of the black-white pebbling rules that a white pebble can be removed if all predecessors are pebbled and that a black pebble can always be removed. The other two rules are harder to understand in terms of black-white pebbling. In particular, the weakening rule allowing us to place black pebbles anywhere might seem somewhat dangerous, and similarly the erasure rule allowing us to remove any white pebbles at any time. It is not hard to see that relaxing the rules in standard black-white pebbling in this way destroys the pebble game completely.

However, the rules make more sense from a resolution derivation perspective. If we think of black pebbles as truth, white pebbles as falsity, and subconfiguration as disjunctions, then the download rule seems to correspond to axiom download in resolution. The resolution rule corresponds to resolving two clauses, and the condition $B_1 \cap W_2 = \emptyset$ then means simply that the resolvent should not contain both a literal and its negation and thus be trivially true. The weakening rule corresponds to weakening in resolution, i.e., deriving a weaker statement that something that we already know, and again the condition $(B \cup B') \cap (W \cup W') = \emptyset$ just says that the result should not be something trivially true. Finally, erasure is simply the parallel of erasure of a clause.

So far, the game can be seen to be in essence a variant of the blob-pebble game in Chapter 9, albeit with some nice simplifications, but a fundamental difference is that the cost of a pebble configuration $\mathbb{R}$ is now $\mathit{cost}(\mathbb{R}) = \left|\bigcup_{[B_i]\langle W_i\rangle\in\mathbb{R}}(B_i \cup W_i)\right|$, i.e., we can charge for every pebbled vertex again. We define the resolution-pebbling price $\mathit{Res\text{-}Peb}(G)$ of G to be the cheapest pebbling of G reaching $\{[z]\langle\emptyset\rangle\}$.

Using this resolution-pebbling game, we construct a lower bound proof as outlined in Section 10.1.1. First, we establish that any resolution refutation of a XOR-pebbling contradiction can be interpreted as a res-pebbling on the DAG in terms of which this formula is defined. Intuitively, the reason that this works is that we can use the weakening rule to analyze apparently non-optimal steps in the refutation.

Theorem 10.1. *Let $Peb^d_G[\oplus]$ denote the XOR-pebbling contradiction of degree $d \geq 1$ over a DAG G with a unique sink z and all non-source vertices having indegree 2.*

Then there is a translation function from sets of clauses derived from $Peb^d_G[\oplus]$ into sets of subconfigurations $\big\{[B_i]\langle W_i\rangle \big| i = 1,\ldots,m\big\}$ such that any resolution refutation π of $Peb^d_G[\oplus]$ corresponds to a res-pebbling $\mathcal{R}_\pi$ of G under this translation.

The translation function is basically that $\mathbb{C}$ corresponds to all $[B]\langle W\rangle$ such that

$$\mathbb{C} \vDash \bigvee_{b\in B} \bigoplus\nolimits_{i=1}^{d} b_i \vee \bigvee_{w\in W} \neg\bigoplus\nolimits_{i=1}^{d} w_i \tag{10.3}$$

and such that this implication does not hold for any strict subset $B' \subsetneqq B$ or $W' \subsetneqq W$. Using this translation of clauses into pebbles, we prove that the clause configurations $\mathbb{C}_0, \mathbb{C}_1, \ldots, \mathbb{C}_\tau$ in a resolution derivation π correspond to “snapshots” at different time intervals of a res-pebbling $\mathcal{R}_\pi$ of the DAG G. Furthermore, we show that the cost of this pebbling is essentially upper-bounded by the largest cost we see at any of the snapshots. There may be many pebbling moves needed to go from the pebble configuration corresponding to $\mathbb{C}_t$ to the one corresponding to $\mathbb{C}_{t+1}$, but the maximal cost during this intermediate pebbling moves is at most an additive constant larger than the cost of the pebble configuration corresponding to $\mathbb{C}_t$ or $\mathbb{C}_{t+1}$. Therefore, the cost of the res-pebbling $\mathcal{R}_\pi$ yields a lower bound on the space of the resolution refutation π.

Theorem 10.2. *If π is a refutation of a XOR-pebbling contradiction $Peb^d_G[\oplus]$ of fixed degree $d > 1$, then the cost of the associated res-pebbling $\mathcal{P}_\pi$ is bounded by the space of π by $\mathsf{cost}(\mathcal{R}_\pi) \leq Sp(\pi) + \mathrm{O}(1)$.*

This is the place in the proof where it is crucial that we are working with XOR-pebbling contradictions $Peb^d_G[\oplus]$ and not standard pebbling contradictions $Peb^d_G[\vee]$ using logical or. In the blob-pebble game in Chapter 9, no matter how large B is we can only charge (at most) 1 for B, since only one clause is needed to imply $\bigvee_{b\in B} \bigvee_{i=1}^{d} b_i$. However, if $\mathbb{C} \vDash \bigvee_{b\in B} \bigoplus_{i=1}^{d} b_i$ and $\mathbb{C}$ does not imply the XOR of any strict subset of B, then it is not hard to show that $|\mathbb{C}| \geq |B|$. And this linear lower bound can be generalized to any set of subconfigurations $[B_1]\langle W_1\rangle, \ldots, [B_m]\langle W_m\rangle$ no matter how the $[B_i]\langle W_i\rangle$ intersect with one another.

Finally, we need lower bounds on res-pebbling price. Since the resolution-pebbling game is a different game than standard black-white pebbling, known bound on black-white pebbling price in the literature do not apply. However, by showing that a a res-pebbling can never do better than a standard black-white pebbling, we can nevertheless reduce res-pebbling price to black-white pebbling price.

Theorem 10.3. *For any DAG G with a unique sink z and all non-source vertices having fan-in 2, it holds that $\mathsf{Res\text{-}Peb}(G) \geq \mathsf{BW\text{-}Peb}(G)$.*

On the face of it, the resolution-pebbling game might seem quite different from the standard black-white pebble game. The lower bounds on black-white pebbling depend critically on the fact that the rules for black pebble placement and white pebble removal are very strict. In the resolution-pebbling game, however, we can always remove any white pebbles by doing an erasure, and by weakening we can always black-pebble any vertex although no white pebbles are not even near this vertex. However, the fact that we collect black pebbles B and white pebbles W in subconfigurations $[B]\langle W\rangle$, and only allow operations on these subconfigurations, makes it relatively straightforward to show Theorem 10.3. The proof hinges on two observations:

1. Given any res-pebbling $\mathcal{R}$ using weakening, we can always find another res-pebbling $\mathcal{R}'$ which is at least as cheap and never makes any weakening moves.
2. The res-pebbling game without the weakening rule is in effect just a disguised version of the standard black-white pebble game.

Putting all of this together, we can prove our main theorem.

Theorem 2.5 (restated). *Let $\mathit{Peb}^d_G[\oplus]$ denote the pebbling contradiction of fixed degree $d > 1$ defined over a DAG G with a unique sink z and all non-source vertices having fan-in 2. Then the clause space of refuting $\mathit{Peb}^d_G[\oplus]$ by resolution is $\mathit{Sp}(\mathit{Peb}^d_G[\oplus] \vdash 0) \geq \mathsf{BW\text{-}Peb}(G) - \mathrm{O}(1)$.*

Proof. Let π be any resolution refutation of $\mathit{Peb}^d_G[\oplus]$ and consider the associated res-pebbling $\mathcal{R}_\pi$ provided by Theorem 10.1. On the one hand, we know that $\mathsf{cost}(\mathcal{R}_\pi) \leq \mathit{Sp}(\pi) + \mathrm{O}(1)$ by Theorem 10.2, provided that $d > 1$. On the other hand, Theorem 10.3 tells us that the cost of any res-pebbling of G is $\mathsf{BW\text{-}Peb}(G)$, so, in particular, we must have $\mathsf{cost}(\mathcal{R}_\pi) \geq \mathsf{BW\text{-}Peb}(G)$. Combining these two bounds on $\mathsf{cost}(\mathcal{R}_\pi)$, we see that $\mathit{Sp}(\pi) \geq \mathsf{BW\text{-}Peb}(G) - \mathrm{O}(1)$. □

With a minor additional effort, we get the separation between space and length as a corollary.

Corollary 2.6 (restated). *For all $k \geq 6$ there is a family $\{F_n\}_{n=1}^{\infty}$ of k-CNF formulas of size $\mathrm{O}(n)$ such that $L(F_n \vdash 0) = \mathrm{O}(n)$ and $W(F_n \vdash 0) = \mathrm{O}(1)$ but $\mathit{Sp}(F_n \vdash 0) = \Omega(n/\log n)$.*

Proof sketch. Let $\{G_n\}_{n=1}^{\infty}$ be the family of DAGs in Theorem 5.5 with size $\mathrm{O}(n)$ and pebbling price $\mathsf{BW\text{-}Peb}(G_n) = \Omega(n/\log n)$. Then if we fix $d \geq 2$ to some constant and set $F_n = \mathit{Peb}^d_{G_n}[\oplus]$, we get a family of $3d$-CNF formulas of size $\mathrm{O}(n)$ that can be refuted in length $L(F_n \vdash 0) = \mathrm{O}(n)$ and width $W(F_n \vdash 0) = \mathrm{O}(1)$ but according to Theorem 2.5 require clause space $\mathit{Sp}(F_n \vdash 0) = \Omega(n/\log n)$. Proving a slightly more general version of the theorem, we can get the same result for k-CNF formulas for any $k \geq 6$. □

10.2 Generalized Pebbling Contradictions

In this section we present the generalization of the formulas in Definition 5.6 that we use to prove the optimal separation of space and length. In order to make the formal definition, we first need to introduce some auxiliary notation.

For any non-constant Boolean function $f_d : \{0,1\}^d \mapsto \{0,1\}$ we fix some canonical representation of it as a CNF formula. We will use the notation $\vec{x} = \{x_1, \ldots, x_d\}$, where d is assumed to be clear from context, so that $f_d(\vec{x})$ is another way of writing $f_d(x_1, \ldots, x_d)$. We let $\mathit{Cl}[f_d(\vec{x})]$ denote the set of clauses in the canonical representation of f_d and $\mathit{Cl}[\neg f_d(\vec{x})]$ denote the clauses in the canonical representation of its negation. For instance, we have

$$\mathit{Cl}[\vee_2(\vec{x})] = \{x_1 \vee x_2\} \quad \text{and} \quad \mathit{Cl}[\neg\vee_2(\vec{x})] = \{\overline{x}_1, \overline{x}_2\} \tag{10.4}$$

for logical or and

$$\mathit{Cl}[\oplus_2(\vec{x})] = \{x_1 \vee x_2, \overline{x}_1 \vee \overline{x}_2\} \quad \text{and} \quad \mathit{Cl}[\neg\oplus_2(\vec{x})] = \{x_1 \vee \overline{x}_2, \overline{x}_1 \vee x_2\} \tag{10.5}$$

for logical exclusive or. The general definitions for exclusive or are

$$\mathit{Cl}[\oplus_d(\vec{x})] = \left\{\textstyle\bigvee_{i=1}^{d} x_i^{\nu_i} \,\middle|\, \sum_{i=1}^{d} \nu_i \equiv d \pmod{2}\right\} \tag{10.6}$$

and

$$\mathit{Cl}[\neg\oplus_d(\vec{x})] = \left\{\textstyle\bigvee_{i=1}^{d} x_i^{\nu_i} \,\middle|\, \sum_{i=1}^{d} \nu_i \not\equiv d \pmod{2}\right\} \tag{10.7}$$

from which we can see that $\mathit{Cl}[\oplus_d(\vec{x})]$ and $\mathit{Cl}[\neg\oplus_d(\vec{x})]$ both are d-CNFs. We will also be interested in the function saying that k out of d variables are true, which we will denote k-true_d. To give an example, for 2-true_4 we have

$$\mathit{Cl}[2\text{-}\mathit{true}_4(\vec{x})] = \begin{Bmatrix} x_1 \vee x_2 \vee x_3, \\ x_1 \vee x_2 \vee x_4, \\ x_1 \vee x_3 \vee x_4, \\ x_2 \vee x_3 \vee x_4 \end{Bmatrix} \tag{10.8}$$

and

$$\mathit{Cl}[\neg 2\text{-}\mathit{true}_4(\vec{x})] = \begin{Bmatrix} \overline{x}_1 \vee \overline{x}_2, \\ \overline{x}_1 \vee \overline{x}_3, \\ \overline{x}_1 \vee \overline{x}_4, \\ \overline{x}_2 \vee \overline{x}_3, \\ \overline{x}_2 \vee \overline{x}_4, \\ \overline{x}_3 \vee \overline{x}_4 \end{Bmatrix} \tag{10.9}$$

and in general we have

$$\mathit{Cl}[k\text{-}\mathit{true}_d(\vec{x})] = \left\{\textstyle\bigvee_{i \in S} x_i \,\middle|\, S \subseteq [d],\ |S| = d - k + 1\right\} \tag{10.10}$$

and

$$\mathit{Cl}[\neg k\text{-}\mathit{true}_d(\vec{x})] = \left\{\textstyle\bigvee_{i \in S} \overline{x}_i \,\middle|\, S \subseteq [d],\ |S| = k\right\} . \tag{10.11}$$

Clearly, 1-$\mathit{true}_d(x_1, \ldots, x_d)$ is just another way of writing the function $\bigvee_{i=1}^{d} x_i$, and d-$\mathit{true}_d(x_1, \ldots, x_d) = \bigwedge_{i=1}^{d} x_i$.

For convenience of notation, we also define the disjunction $\mathbb{C} \lor \mathbb{D}$ of two clause sets $\mathbb{C}$ and $\mathbb{D}$ to be the clause set

$$\mathbb{C} \lor \mathbb{D} = \{C \lor D \mid C \in \mathbb{C}, \, D \in \mathbb{D}\} \ . \tag{10.12}$$

This notation extends to more than two clause sets in the natural way. Using the notation above, we define pebbling contradictions as follows.

Definition 10.4 (Generalized pebbling contradiction). Suppose that G is a DAG with sources S, a unique sink z and with all non-source vertices having indegree 2, that $d > 0$ is an integer, and that $f_d : \{0,1\}^d \mapsto \{0,1\}$ is any non-constant Boolean function. Associate d distinct variables $v_1, \ldots, v_d$ with every vertex $v \in V(G)$. The dth degree *pebbling contradiction* over G with respect to f_d, denoted $\mathit{Peb}^d_G[f_d]$, is the conjunction over the following clauses:

- $\mathit{Cl}[f_d(\vec{s})]$ for all $s \in S$ (*source axioms*),
- $\mathit{Cl}[\neg f_d(\vec{u})] \lor \mathit{Cl}[\neg f_d(\vec{v})] \lor \mathit{Cl}[f_d(\vec{w})]$ for all $w \in V(G) \setminus S$, where u, v are the two predecessors of w (*pebbling axioms*),
- $\mathit{Cl}[\neg f_d(\vec{z})]$ for the sink z (*target* or *sink axioms*).

In general, the formula $\mathit{Peb}^d_G[f_d]$ is an unsatisfiable CNF formula of width at most $3d$ with at most $\mathrm{O}\bigl(2^{3d} \cdot |V(G)|\bigr)$ clauses over $d \cdot |V(G)|$ variables. To be slightly more precise, the width of the formula is easily seen to be $W\bigl(\mathit{Peb}^d_G[f_d]\bigr) = 2 \cdot W(\mathit{Cl}[\neg f_d(\vec{x})]) + W(\mathit{Cl}[f_d(\vec{x})])$.

When the function f_d comes from some general family of functions defined for all arities, as will be the case in the rest of this chapter, we will often skip the arity subscript and write just $\mathit{Peb}^d_G[f]$. We will refer to the formulas $\mathit{Peb}^d_G[\oplus]$ as *XOR-pebbling contradictions* and the formulas $\mathit{Peb}^d_G[k\text{-}true]$ as *k-true-pebbling contradictions*. It is easy to verify that the width of k-true-pebbling contradictions is $W\bigl(\mathit{Peb}^d_G[k\text{-}true]\bigr) = d + k + 1$.

Proposition 10.5. *For any G as in Definition 10.4, if d is constant there is a refutation $\pi : \mathit{Peb}^d_G[f] \vdash 0$ in length $L(\pi) = \mathrm{O}(|V(G)|)$ and width $W(\pi) = \mathrm{O}(1)$. Also, $\mathit{Peb}^d_G[f]$ is refutable in clause space $\mathit{Sp}\bigl(\mathit{Peb}^d_G[f] \vdash 0\bigr) = \mathrm{O}\bigl(\mathsf{Peb}(G)\bigr)$.*

Proof. This is just a generalization of the proofs of Propositions 5.7 and 5.10. Given any black pebbling of G, we construct a resolution refutation of $\mathit{Peb}^d_G[f]$ such that if at some point in time there are black pebbles on a set of vertices V, then we have the clauses $\bigcup_{v \in V} \mathit{Cl}[f_d(\vec{v})]$ in memory. When some new vertex v is pebbled, we derive $\mathit{Cl}[f_d(\vec{v})]$ using the axiom clauses for v plus the clauses already in memory. We claim that this can be done in constant extra space if d is fixed. When a black pebble is removed from v, we erase the clauses $\mathit{Cl}[f_d(\vec{v})]$. We conclude the resolution refutation by resolving $\mathit{Cl}[f_d(\vec{z})]$ for the sink z with all sink axioms $\mathit{Cl}[\neg f_d(\vec{z})]$, which can also be done in constant space. It is clear that given our claims about the constant extra space needed, this yields a resolution refutation in space linear in the pebbling cost. In particular, given an optimal black pebbling of G, we get a refutation in space $\mathrm{O}\bigl(\mathit{Peb}(G)\bigr)$.

To prove the claim, note first that it trivially holds for source vertices v, since $\mathit{Cl}[f_d(\vec{v})]$ is a set of axiom clauses of size $\mathrm{O}\bigl(2^d\bigr) = \mathrm{O}(1)$. Suppose for a non-source vertex r with predecessors p and q that at some point in time a black pebble is placed on r. Then p and q must be black-pebbled, so by induction we have the clauses $\mathit{Cl}[f_d(\vec{p})]$ and $\mathit{Cl}[f_d(\vec{q})]$ in memory. Download all pebbling axioms $\mathit{Cl}[\neg f_d(\vec{p})] \vee \mathit{Cl}[\neg f_d(\vec{q})] \vee \mathit{Cl}[f_d(\vec{r})]$ for r. Then

$$\mathit{Cl}[f_d(\vec{p})] \cup \mathit{Cl}[f_d(\vec{q})] \cup \mathit{Cl}[\neg f_d(\vec{p})] \vee \mathit{Cl}[\neg f_d(\vec{q})] \vee \mathit{Cl}[f_d(\vec{r})] \vDash \mathit{Cl}[f_d(\vec{r})] \tag{10.13}$$

and since resolution is sound this means that we can derive $\mathit{Cl}[f_d(\vec{r})]$ from the clauses on the left-hand side of (10.13). Furthermore, since the clauses contain $3d$ variables, this subderivation can be performed in length at most $\mathrm{O}\bigl(2^{3d}\bigr)$, width at most $\mathrm{O}(d)$, and space at most $\mathrm{O}\bigl(2^{3d}\bigr)$, which are all constant if d is constant.

To get the statements for length and width, consider a pebbling that black-pebbles all vertices once in topological order without ever removing a pebble. This yields a refutation in length $L(\pi) = \mathrm{O}\bigl(2^{3d} \cdot |V(G)|\bigr)$ and width $\mathrm{O}(d)$, i.e., in linear length and constant width when d is fixed. □

The following observation is rather immediate, but nevertheless it might be helpful to state it explicitly.

Observation 10.6. *Suppose for any non-constant Boolean function f_d that $C \in \mathit{Cl}[f_d(\vec{x})]$ and that ρ is any partial truth value assignment such that $\rho(C) = 0$. Then for all $D \in \mathit{Cl}[\neg f_d(\vec{x})]$ it holds that $\rho(D) = 1$.*

Proof. If $\rho(C) = 0$ this means that $\rho(f_d) = 0$. Then clearly $\rho(\neg f_d) = 1$, so, in particular, ρ must fix all clauses $D \in \mathit{Cl}[\neg f_d(\vec{x})]$ to true. □

10.3 The Resolution-Pebbling Game

In this section we define our modified pebble game and show that the pebbling price in this game is lower-bounded by the standard black-white pebbling price. In the following, we will almost exclusively discuss DAGs with a unique sink and with all vertices having indegree 0 or 2. We will refer to such DAGs as *single-sink fan-in* 2 *DAGs* for brevity. From now on, by "DAG" we mean "single-sink fan-in 2 DAG" unless stated otherwise.

The resolution-pebbling game is in a sense fairly similar to the blob-pebble game in Chapter 9, but in contrast to the blob-pebble game we do not place any restrictions on what B can look like or where the associated white pebbles W can be located relative to B.

Definition 10.7 (Res-pebbling subconfiguration). If B and W are sets of vertices in a single-sink fan-in 2 DAG G with $B \neq \emptyset$, $B \cap W = \emptyset$, we say that $[B]\langle W\rangle$

is a *res-pebbling subconfiguration*, or just *subconfiguration*, in G with black pebbles on B and white pebbles on W *supporting* B. We also say that B is *dependent* on W. If $W = \emptyset$, we say that B is *independent*. A set of subconfigurations $\mathbb{R} = \big\{[B_i]\langle W_i\rangle \big| i = 1, \ldots, m\big\}$ is a *res-pebbling configuration.*

The simplification of the definition of subconfigurations also makes the pebble game less complicated compared to Chapter 9.

Definition 10.8 (Resolution-pebbling game). For G a single-sink fan-in 2 DAG G, a *res-pebbling* from $\mathbb{R}_0$ to $\mathbb{R}_\tau$ in G is a sequence $\mathcal{R} = \big\{\mathbb{R}_0, \ldots, \mathbb{R}_\tau\big\}$ of res-pebbling configurations such that for $t \in [\tau]$, $\mathbb{R}_t$ is obtained from $\mathbb{R}_{t-1}$ by one of the following rules:

Download $\mathbb{R}_t = \mathbb{R}_{t-1} \cup \{[v]\langle pred(v)\rangle\}$ for any vertex v.

Resolution $\mathbb{R}_t = \mathbb{R}_{t-1} \cup \big\{[B_1 \cup B_2]\langle W_1 \cup W_2\rangle\big\}$ if there are subconfigurations $[B_1]\langle W_1 \cup \{v\}\rangle, [B_2 \cup \{v\}]\langle W_2\rangle \in \mathbb{R}_{t-1}$ such that $B_1 \cap W_2 = \emptyset$.

Weakening $\mathbb{R}_t = \mathbb{R}_{t-1} \cup \big\{[B \cup B']\langle W \cup W'\rangle\big\}$ if $[B]\langle W\rangle \in \mathbb{R}_{t-1}$ and $(B \cup B') \cap (W \cup W') = \emptyset$.

Erasure $\mathbb{R}_t = \mathbb{R}_{t-1} \setminus \big\{[B]\langle W\rangle\big\}$ for $[B]\langle W\rangle \in \mathbb{R}_{t-1}$.

A *complete res-pebbling* of G is a res-pebbling $\mathcal{R}$ with $\mathbb{R}_0 = \emptyset$ and $\mathbb{R}_\tau = \big\{[z]\langle\emptyset\rangle\big\}$ for z the unique sink of G.

The *cost* of a pebble configuration $\mathbb{R}$ is $\mathsf{cost}(\mathbb{R}) = \big|\bigcup_{\mathbb{R}B_iW_i \in \mathbb{R}}(B_i \cup W_i)\big|$ and the cost of a res-pebbling $\mathcal{R} = \big\{\mathbb{R}_0, \ldots, \mathbb{R}_\tau\big\}$ is $\mathsf{cost}(\mathcal{R}) = \max_{t\in[\tau]}\{\mathsf{cost}(\mathbb{R}_t)\}$. The resolution-pebbling price $\mathsf{Res\text{-}Peb}(G)$ of G is the cheapest complete pebbling of G.

We now prove that no resolution-pebbling can make a cheaper pebbling of a DAG G than an optimal black-white pebbling.

Theorem 10.3 (restated). *For any single-sink fan-in* 2 *DAG G it holds that* $\mathsf{Res\text{-}Peb}(G) \geq \mathsf{BW\text{-}Peb}(G)$.

This theorem follows immediately from the following two lemmas.

Lemma 10.9. *Given any complete res-pebbling $\mathcal{R}$ of G using weakening, there is a complete res-pebbling $\mathcal{R}'$ which never makes any weakening moves and has* $\mathsf{cost}(\mathcal{R}') \leq \mathsf{cost}(\mathcal{R})$.

Lemma 10.10. *Given any complete res-pebbling $\mathcal{R}'$ of G that does not make any weakening moves, there is a complete standard black-white pebbling $\mathcal{P}$ of G such that* $\mathsf{cost}(\mathcal{P}) \leq \mathsf{cost}(\mathcal{R}')$.

Proof of Lemma 10.9. We construct a shadow pebbling that matches download, resolution, and erasure moves but ignores weakening moves. Such a pebbling can have at most the same cost as the pebbling that it is shadowing.

Formally, given any complete res-pebbling $\mathcal{R} = \{\mathbb{R}_0, \ldots, \mathbb{R}_\tau\}$ of G, we construct our pebbling $\mathcal{R}' = \{\mathbb{R}'_0, \ldots, \mathbb{R}'_\tau\}$ inductively by maintaining the following invariant: For every $\mathbb{R}_t \in \mathcal{R}$ there is a surjective function $g_t : \mathbb{R}_t \mapsto \mathbb{R}'_t$ such that for $g_t([B]\langle W\rangle) = [b]\langle W'\rangle$ it holds that $b \in B$ and $W' \subseteq W$. If we can construct such a function g_t for every t we are clearly done, since $\mathit{cost}(\mathbb{R}'_t) = \mathit{cost}(g_t(\mathbb{R}_t)) \leq \mathit{cost}(\mathbb{R}_t)$ and we must have $g_\tau([z]\langle\emptyset\rangle) = \{[z]\langle\emptyset\rangle\}$. The base case $\mathbb{R}_0 = \emptyset$ is trivial. We make a case analysis over the pebbling move made at time t.

Download $\mathbb{R}_t = \mathbb{R}_{t-1} \cup \{[v]\langle \mathit{pred}(v)\rangle\}$: Make the same download move in $\mathcal{R}'$, set $g_t([v]\langle \mathit{pred}(v)\rangle) = [v]\langle \mathit{pred}(v)\rangle$ and let $g_t = g_{t-1}$ for all other subconfigurations in $\mathbb{R}_{t-1}$.

Erasure $\mathbb{R}_t = \mathbb{R}_{t-1} \setminus \{[B]\langle W\rangle\}$: Set $\mathbb{R}'_t = g_{t-1}(\mathbb{R}_t)$ (which might result in an erasure or leave $\mathbb{R}'_t = \mathbb{R}'_{t-1}$ unchanged).

Weakening $\mathbb{R}_t = \mathbb{R}_{t-1} \cup \{[B \cup B']\langle W \cup W'\rangle\}$ for some $[B]\langle W\rangle \in \mathbb{R}_{t-1}$: set $g_t([B \cup B']\langle W \cup W'\rangle) = g_{t-1}([B]\langle W\rangle)$ and let $g_t = g_{t-1}$ for all other subconfigurations (leaving $\mathbb{R}'_t = \mathbb{R}'_{t-1}$ unchanged).

Resolution $\mathbb{R}_t = \mathbb{R}_{t-1} \cup \{[B_1 \cup B_2]\langle W_1 \cup W_2\rangle\}$ derived from $[B_1]\langle W_1 \cup \{v\}\rangle$ and $[B_2 \cup \{v\}]\langle W_2\rangle$ in $\mathbb{R}_{t-1}$: This is the only nontrivial case. Let us write $[b_1]\langle W'_1\rangle = g_{t-1}([B_1]\langle W_1 \cup \{v\}\rangle)$ and $[b_2]\langle W'_2\rangle = g_{t-1}([B_2 \cup \{v\}]\langle W_2\rangle)$. We note that by the induction hypothesis we have $b_1 \in B_1 \subseteq B_1 \cup B_2$ and $W'_2 \subseteq W_2 \subseteq W_1 \cup W_2$. There are three subcases to consider:

1. $v \notin W'_1$: Then it holds that $W'_1 \subseteq W_1 \subseteq W_1 \cup W_2$, so we can set $g_t([B_1 \cup B_2]\langle W_1 \cup W_2\rangle) = [b_1]\langle W'_1\rangle$.
2. $v \neq b_2$: In this case we have $b_2 \in B_2 \subseteq B_1 \cup B_2$ and we can define $g_t([B_1 \cup B_2]\langle W_1 \cup W_2\rangle) = [b_2]\langle W'_2\rangle$.
3. If none of the above two cases hold, we have $v = b_2$ and $v \in W'_1$, so we can resolve $[b_1]\langle W'_1\rangle$ and $[b_2]\langle W'_2\rangle$ to get $[b_1]\langle (W'_1 \cup W'_2) \setminus \{b_2\}\rangle$ and set $g_t([B_1 \cup B_2]\langle W_1 \cup W_2\rangle) = [b_1]\langle (W'_1 \cup W'_2) \setminus \{b_2\}\rangle$.

Let $g_t = g_{t-1}$ for all other subconfigurations in $\mathbb{R}_{t-1}$.

Since in all cases we can construct a surjective function $g_t : \mathbb{R}_t \mapsto \mathbb{R}'_t$ satisfying the invariant conditions, the lemma follows. □

Proof of Lemma 10.10. This second lemma holds since the res-pebbling game without weakening moves is just a poorly disguised version of the black-white pebble game. The erasure rule might seem to allow uncontrolled white pebble removal, but without loss of generality a subconfiguration $[B]\langle W\rangle$ is erased precisely when it has been used for the last time in a resolution move, and if so the net change of white pebbles on G is at most 1. This white pebbles on, say, w has been removed only if there is currently a black pebble on w. But since this can only happen if all predecessors of w either are pebbled now or have been pebbled before, this does not really change anything compared to the standard black-white pebbling rule for white pebble removal. The formal details of the proof, which are straightforward if somewhat tedious, can be found in the proof of Lemma 8.30. (The labelled pebble game in Definition 8.5 without reversal moves is easily seen to be identical to the resolution-pebbling game without weakening moves.) $\square$

10.4 Derivations Induce Resolution-Pebblings

The next step in our construction is to show that resolution refutations can be interpreted in terms of resolution-pebblings. As in Chapters 8 and 9, we get a cleaner correspondence between resolution and res-pebblings if we ignore the sink axioms $Cl[\neg f_d(\vec{z})]$ and instead study resolution derivations of $Cl[f_d(\vec{z})]$ from the rest of the formula rather than resolution refutations of all of $Peb^d_G[f]$.

Let us write ${}^*Peb^d_G[f] = Peb^d_G[f] \setminus Cl[\neg f_d(\vec{z})]$ to denote the pebbling formula over G with the target axioms in the pebbling contradiction removed. The next lemma is the formal statement saying that as long as we keep the pebbling degree d constant, we may just as well study resolution derivations of $Cl[f_d(\vec{z})]$ from ${}^*Peb^d_G[f]$ instead of refutations of $Peb^d_G[f]$ without losing more than a constant term.

Lemma 10.11. *For any single-sink fan-in* 2 *DAG G with sink z, it holds that* $Sp(Peb^d_G[f] \vdash 0) = Sp({}^*Peb^d_G[f] \vdash Cl[f_d(\vec{z})]) + \mathrm{O}(2^d)$.

Proof. For any resolution derivation $\pi^* : {}^*Peb^d_G[f] \vdash Cl[f_d(\vec{z})]$, we can get a refutation of $Peb^d_G[f]$ from π^* in at most $\mathrm{O}(2^d)$ extra space by downloading all target axioms $Cl[\neg f_d(\vec{z})]$ and then, keeping all clauses in memory, deriving the empty clause in additional space $d + \mathrm{O}(1)$ (since any formula over n variables is refutable in space $n + \mathrm{O}(1)$ by [39]).

In the other direction, suppose we have a refutation $\pi : Peb^d_G[f] \vdash 0$. Consider the restricted refutations $\pi\restriction_{\rho(\neg C)}$ for all $C \in Cl[f_d(\vec{z})]$. These restrictions satisfy all sink axioms by Observation 10.6, so these axioms are never used in the restricted resolution refutations $\pi\restriction_{\rho(\neg C)}$. Also, by Proposition 4.17 these restricted refutations all have space at most $Sp(\pi)$. Removing the restrictions again, this means that we get resolution derivations $\pi_C : {}^*Peb^d_G[f] \vdash C$ for all $C \in Cl[f_d(\vec{z})]$ (we must get derivations of some $C' \subseteq C$, and since resolution is sound we must have $C' = C$). There are at most 2^d clauses in $Cl[f_d(\vec{z})]$, and by performing all derivations π_C, $C \in Cl[f_d(\vec{z})]$, one after another and saving all final clauses C in memory, we get a derivation $\pi^* : {}^*Peb^d_G[f] \vdash Cl[f_d(\vec{z})]$ in space at most $Sp(\pi) + 2^d$. $\square$

In view of Lemma 10.11, from now on we will only consider resolution derivations from ${}^*Peb^d_G[f]$ and try to convert clause configurations in such derivations into sets of res-pebbling subconfigurations. Note that since ${}^*Peb^d_G[f]$ is non-contradictory and resolution is sound, this means that any clause set $\mathbb{C}$ derived from ${}^*Peb^d_G[f]$ is satisfiable.

To avoid cluttering the notation with an excessive amount of brackets, we will use sloppy notation for sets. We will sometimes omit curly brackets around singleton sets when no confusion can arise, writing, for instance, $V \cup v$ instead of $V \cup \{v\}$ and $[B \cup b]\langle W \cup w\rangle$ instead of $[B \cup \{b\}]\langle W \cup \{w\}\rangle$. Also, we will sometimes omit the curly brackets around sets of vertices in $[B]\langle W\rangle$ when we enumerate their members, writing, for instance, $[b_1, b_2, b_3]\langle w_1, w_2, w_3\rangle$ instead of $[\{b_1, b_2, b_3\}]\langle\{w_1, w_2, w_3\}\rangle$.

10.4.1 Definition of Induced Configurations and Theorem Statement

For the rest of this section, let $f(x_1, \ldots, x_d)$ be some arbitrary but fixed (non-constant) Boolean function. If r is a non-source vertex with predecessors $\mathit{pred}(r) = \{p, q\}$, we say that the *axioms for* r in $*\mathit{Peb}^d_G[f]$ are

$$Ax^d(r) = \mathit{Cl}[\neg f_d(\vec{p})] \vee \mathit{Cl}[\neg f_d(\vec{q})] \vee \mathit{Cl}[f_d(\vec{r})] \tag{10.14}$$

where we recall that $\mathit{Cl}[\neg f_d(\vec{p})] \vee \mathit{Cl}[\neg f_d(\vec{q})] \vee \mathit{Cl}[f_d(\vec{r})]$ is the set of clauses

$$\left\{C_{\neg p} \vee C_{\neg q} \vee C_r \,\middle|\, C_{\neg p} \in \mathit{Cl}[\neg f_d(\vec{p})],\, C_{\neg q} \in \mathit{Cl}[\neg f_d(\vec{q})],\, C_r \in \mathit{Cl}[f_d(\vec{r})]\right\} \ , \tag{10.15}$$

and if r is a source, we define

$$Ax^d(r) = \mathit{Cl}[f_d(\vec{r})] \ . \tag{10.16}$$

For U a set of vertices in G, we let $Ax^d(U) = \bigcup_{u \in U} Ax^d(u)$. Note that with this notation, we have $*\mathit{Peb}^d_G[f] = \bigcup_{v \in V(G)} Ax^d(v)$.

Recall that we say that a set of clauses $\mathbb{C}$ implies a clause D *minimally* if $\mathbb{C} \vDash D$ but for all $\mathbb{C}' \subsetneqq \mathbb{C}$ it holds that $\mathbb{C}' \nvDash D$. We say that $\mathbb{C}$ implies a clause D *maximally* if $\mathbb{C} \vDash D$ but for all $D' \subsetneqq D$ it holds that $\mathbb{C}' \nvDash D'$. To define our translation of clauses to res-pebbling subconfigurations, we use implications that are in a sense both minimal and maximal. The following definition is similar in spirit to that in Chapter 9, but again it is simpler thanks to the fact that we have a simpler pebble game.

Definition 10.12 (Induced res-pebbling subconfiguration). Let G be a DAG and $\mathbb{C}$ a set of clauses derived from $*\mathit{Peb}^d_G[f]$. Then $\mathbb{C}$ induces the res-pebbling subconfiguration $[B]\langle W\rangle$ if there is a clause set $\mathbb{D} \subseteq \mathbb{C}$ such that

$$\mathbb{D} \vDash \bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w}) \tag{10.17a}$$

but for which it holds for all strict subsets $\mathbb{D}' \subsetneqq \mathbb{D}$, $B' \subsetneqq B$ and $W' \subsetneqq W$ that

$$\mathbb{D}' \nvDash \bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w}) \ , \tag{10.17b}$$

$$\mathbb{D} \nvDash \bigvee_{b \in B'} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w}) \ , \text{ and} \tag{10.17c}$$

$$\mathbb{D} \nvDash \bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W'} \neg f_d(\vec{w}) \ . \tag{10.17d}$$

To save space, when all conditions (10.17a)–(10.17d) hold, we write

$$\mathbb{D} \rhd \bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w}) \tag{10.18}$$

and refer to this as *precise implication.* We also say that the clause set $\mathbb{D}$ implies $\bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w})$ *precisely.* We write

$$\mathbb{R}(\mathbb{C}) = \left\{ [B]\langle W \rangle \;\middle|\; \exists \mathbb{D} \subseteq \mathbb{C} \text{ s.t. } \mathbb{D} \rhd \bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w}) \right\} \tag{10.19}$$

to denote the set of all res-pebbling subconfigurations induced by $\mathbb{C}$.

The main result of this section is as follows.

Theorem 10.13. *Let $\pi = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ be a resolution derivation of $Cl[f_d(\vec{z})]$ from $^*Peb^d_G[f]$ for some arbitrary non-constant function f. Then the induced res-pebbling configurations $\{\mathbb{R}(\mathbb{C}_0), \ldots, \mathbb{R}(\mathbb{C}_\tau)\}$ form the "backbone" of a complete res-pebbling $\mathcal{R}$ of G in the sense that*

- $\mathbb{R}(\mathbb{C}_0) = \emptyset$,
- $\mathbb{R}(\mathbb{C}_\tau) = \{[z]\langle\emptyset\rangle\}$, *and*
- *for every $t \in [\tau]$, the transition from $\mathbb{R}(\mathbb{C}_{t-1})$ to $\mathbb{R}(\mathbb{C}_t)$ can be accomplished in accordance with the res-pebbling rules in such a way that the intermediate pebbling cost is upper-bounded by $\max\{\mathit{cost}(\mathbb{R}(\mathbb{C}_{t-1})), \mathit{cost}(\mathbb{R}(\mathbb{C}_t))\} + \mathrm{O}(1)$.*

In particular, to any resolution derivation $\pi : {}^*Peb^d_G[f] \vdash Cl[f_d(\vec{z})]$ we can associate a complete res-pebbling $\mathcal{R}_\pi$ of G such that $\mathit{cost}(\mathcal{R}_\pi) \leq \max_{\mathbb{C} \in \pi}\{\mathit{cost}(\mathbb{R}(\mathbb{C}))\} + \mathrm{O}(1)$.

Clearly, Lemma 10.11 and Theorem 10.13 together imply Theorem 10.2.

We prove Theorem 10.13 by forward induction over the derivation π. By the pebbling rules in Definition 10.8, any subconfiguration $[B]\langle W \rangle$ may be erased freely at any time. Consequently, we need not worry about subconfigurations disappearing during the transition from $\mathbb{C}_{t-1}$ to $\mathbb{C}_t$. What we do need to check, though, is that no subconfiguration $[B]\langle W \rangle$ appears inexplicably in $\mathbb{R}(\mathbb{C}_t)$ as a result of a derivation step $\mathbb{C}_{t-1} \rightsquigarrow \mathbb{C}_t$, but that we can always derive any $[B]\langle W \rangle \in \mathbb{R}(\mathbb{C}_t) \setminus \mathbb{R}(\mathbb{C}_{t-1})$ from $\mathbb{R}(\mathbb{C}_{t-1})$ by the res-pebbling rules. Also, when several pebbling moves are needed to get from $\mathbb{R}(\mathbb{C}_t)$ to $\mathbb{R}(\mathbb{C}_{t-1})$, we need to check that these intermediate moves do not affect the pebbling cost by more than an additive constant.

The proof boils down to a case analysis of the different possibilities for the derivation step $\mathbb{C}_{t-1} \rightsquigarrow \mathbb{C}_t$. For clarity, we divide the analysis of the different cases into subsections. But first of all we need some technical results.

10.4.2 Some Easy But Useful Technical Observations

The next two observations are fairly obvious once one deciphers what they say, but they will prove very useful in the proof of Theorem 10.13. We present their proofs for completeness.

Observation 10.14. *If If $\mathbb{C}$ is a clause set derived from *$Peb^d_G[f]$ such that $\mathbb{C} \vDash \bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w})$, then there is a subconfiguration $[B']\langle W'\rangle \in \mathbb{R}(\mathbb{C})$ such that $B' \subseteq B$ and $W' \subseteq W$. In particular, $[B]\langle W\rangle$ is derivable by weakening from $\mathbb{R}(\mathbb{C})$.*

Proof. Just pick any minimal clause set $\mathbb{C}' \subseteq \mathbb{C}$, and any minimal vertex sets $B' \subseteq B$ and $W' \subseteq W$ such that the implication $\mathbb{C}' \vDash \bigvee_{b \in B'} f_d(\vec{b}) \vee \bigvee_{w \in W'} \neg f_d(\vec{w})$ holds. (We note that $B' \neq \emptyset$ since *$Peb^d_G[f] \nvDash \bigvee_{w \in W} \neg f_d(w_1, \ldots, w_d)$ and resolution is sound.) But then by Definition 10.12, it holds that $\mathbb{C}' \rhd \bigvee_{b \in B'} f_d(\vec{b}) \vee \bigvee_{w \in W'} \neg f_d(\vec{w})$, so $[B']\langle W'\rangle \in \mathbb{R}(\mathbb{C})$. □

A particularly interesting application of Observation 10.14 is given in the next observation.

Observation 10.15. *If $\mathbb{C}$ is a clause set derived from *$Peb^d_G[f]$ and $C \in Ax^d(r)$ is an axiom clause for some vertex $r \in V(G)$ such that*

$$\mathbb{C} \cup \{C\} \vDash \bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w}) \ , \tag{10.20}$$

then:

1. *It always holds that $\mathbb{C} \vDash \bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W \,\cup\, \{r\}} \neg f_d(\vec{w})$, so if $r \notin B$ we can derive $[B]\langle W \cup r\rangle$ from $\mathbb{R}(\mathbb{C})$ by weakening.*
2. *If r is a non-source vertex and q is a predecessor of r, then the implication $\mathbb{C} \vDash \bigvee_{b \in B \,\cup\, \{q\}} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w})$ holds. In particular, if $q \notin W$ we can derive $[B \cup q]\langle W\rangle$ from $\mathbb{R}(\mathbb{C})$ by weakening.*

Proof. If $C \in Ax^d(r)$, then by (10.14) and (10.16) there is a subclause $D \subseteq C$ such that $D \in Cl[f_d(\vec{r})]$. Suppose that α is any truth value assignment such that $\alpha(\mathbb{C}) = 1$ but $\alpha\big(\bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w})\big) = 0$ (if there is no such α then we are already done). Then we must have $\alpha(C) = 0$ since otherwise we get a contradiction to (10.20), so in particular $\alpha(D) = 0$. But then $\alpha\big(\neg f_d(r_1, \ldots, r_d)\big) = 1$. Hence, any assignment α that satisfies $\mathbb{C}$ must also satisfy $\bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W \,\cup\, \{r\}} \neg f_d(\vec{w})$. Applying Observation 10.14, we get part 1 above.

Part 2 is very similar. If $C \in Ax^d(r)$ for a non-source vertex r with $q \in pred(r)$, there is a subclause $D \subseteq C$ such that $D \in Cl[\neg f_d(\vec{q})]$ (compare (10.15) above). Let us again pick any truth value assignment α such that $\alpha(\mathbb{C}) = 1$ but $\alpha\big(\bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w})\big) = 0$. Then it must hold that $\alpha(C) = 0$, but this implies that $\alpha(D) = 0$ and $\alpha\big(f_d(q_1, \ldots, q_d)\big) = 1$. □

We are now ready for the case analysis in the proof of Theorem 10.13 for the different possible derivation steps in a resolution derivation.

10.4.3 Erasure

Suppose that $\mathbb{C}_t = \mathbb{C}_{t-1} \setminus \{C\}$ for $C \in \mathbb{C}_{t-1}$. It is easy to see that the only possible outcome of erasing clauses is that res-pebbling subconfigurations disappear. We note for future reference that this implies that the res-pebbling cost decreases monotonically when going from $\mathbb{R}(\mathbb{C}_{t-1})$ to $\mathbb{R}(\mathbb{C}_t)$.

10.4.4 Inference

Suppose that $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C\}$ for some clause C derived from $\mathbb{C}_{t-1}$. No res-pebbling subconfigurations can disappear at an inference move since $\mathbb{C}_{t-1} \subseteq \mathbb{C}_t$. Suppose that $[B]\langle W\rangle$ is a new subconfiguration at time t arising from $\mathbb{D} \subseteq \mathbb{C}_{t-1}$ precisely implying $\bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w})$. Since C is derived from $\mathbb{C}_{t-1}$, we have $\mathbb{C}_{t-1} \vDash C$. Thus it holds that $\mathbb{C}_{t-1} \vDash \bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w})$ and Observation 10.14 tells us that $[B]\langle W\rangle$ is derivable by weakening from $\mathbb{R}(\mathbb{C}_{t-1})$.

Since no subconfiguration disappears, the pebbling cost increases monotonically when going from $\mathbb{R}(\mathbb{C}_{t-1})$ to $\mathbb{R}(\mathbb{C}_t)$ for an inference step, which is again noted for future reference.

10.4.5 Axiom Download

This is the interesting case. Assume that a new res-pebbling subconfiguration $[B]\langle W\rangle$ is induced at time t as the result of a download of an axiom $C \in Ax^d(r)$. Then C must be one of the clauses inducing the subconfiguration, and we get that there is a clause set $\mathbb{C} \subseteq \mathbb{C}_{t-1}$ such that

$$\mathbb{C} \cup \{C\} \rhd \bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w}) \ . \tag{10.21}$$

Our intuition is that download of an axiom clause $C \in Ax^d(r)$ in the resolution derivation should correspond to an introduction of $[r]\langle pred(r)\rangle$ in the induced res-pebbling. We want to prove that any other res-pebbling subconfiguration $[B]\langle W\rangle$ in $\mathbb{R}(\mathbb{C}_t)$ is derivable by the pebbling rules from $\mathbb{R}(\mathbb{C}_{t-1}) \cup \big\{[r]\langle pred(r)\rangle\big\}$. Also, we need to prove that the pebbling moves needed to go from $\mathbb{R}(\mathbb{C}_{t-1})$ to $\mathbb{R}(\mathbb{C}_t)$ do not increase the res-pebbling cost by more than an additive constant compared to $\max\big\{\mathit{cost}(\mathbb{R}(\mathbb{C}_{t-1})), \mathit{cost}(\mathbb{R}(\mathbb{C}_t))\big\} = \mathit{cost}(\mathbb{R}(\mathbb{C}_t))$, where the equality holds since no subconfigurations induced by $\mathbb{C}_{t-1}$ can disappear when we add clauses to $\mathbb{C}_{t-1}$.

As a warm-up, let us consider the case when r is a source, i.e., $pred(r) = \emptyset$ and $C \in Ax^d(r) = Cl[f_d(\vec{r})]$. We make a case analysis depending on whether $r \in B$ in (10.21) or not.

1. $r \in B$: In this case we need no further analysis. Just make the download move $[r]\langle\emptyset\rangle$ and weaken $[r]\langle\emptyset\rangle$ to get $[B \cup r]\langle W\rangle = [B]\langle W\rangle$.
2. $r \notin B$: By part 1 of Observation 10.15, we can derive $[B]\langle W \cup r\rangle$ by weakening from $\mathbb{R}(\mathbb{C}_{t-1})$. Then $[B]\langle W\rangle$ can be derived by a download of $[r]\langle\emptyset\rangle$ followed by a resolution of $[B]\langle W \cup r\rangle$ and $[r]\langle\emptyset\rangle$.

We see that when r is a source, we can get from $\mathbb{R}(\mathbb{C}_{t-1})$ to $\mathbb{R}(\mathbb{C}_t)$ by a download of $[r]\langle\emptyset\rangle$ and possibly some weakenings and resolutions.

The case when r is a non-source is a bit more involved, but the general idea is the same. Suppose for the rest of this section that $C \in Ax^d(r)$ for some fixed vertex r with $pred(r) = \{p, q\}$. This means that C can be written $C = C_{\neg p} \vee C_{\neg q} \vee C_r$ for some $C_{\neg p} \in Cl[\neg f_d(\vec{p})]$, $C_{\neg q} \in Cl[\neg f_d(\vec{q})]$, and $C_r \in Cl[f_d(\vec{r})]$, and we can rewrite (10.21) as

$$\mathbb{C} \cup \left\{C_{\neg p} \vee C_{\neg q} \vee C_r\right\} \rhd \bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w}) \ . \qquad (10.22)$$

Let us also assume that

$$\mathbb{C}_{t-1} \nvDash \bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w}) \qquad (10.23)$$

since otherwise we can derive $[B]\langle W\rangle$ by an weakening move from $\mathbb{R}(\mathbb{C}_{t-1})$ (using Observation 10.14) and be done. Recall that by definition, we have $B \cap W = \emptyset$. Observe that it must hold that

$$\{p, q\} \cap B = \emptyset \ , \qquad (10.24)$$

since if, say, $q \in B$, we could apply part 1 of Observation 10.15 to get that the implication in (10.23) in fact holds for $B = B \cup \{q\}$ contrary to assumption. In the same way, we see that

$$r \notin W \qquad (10.25)$$

since otherwise part 2 of Observation 10.15 shows that the implication (10.23) on the contrary is true for $W = W \cup \{r\}$.

As in the case when r was a source vertex, the induction step is by a case analysis depending on whether or not $r \in B$ in the implication (10.22) (which, we remind ourselves, is just (10.21) with added information about what the downloaded axiom clause C looks like).

1. $r \in B$: We split this case into subcases depending on whether $p, q \in W$ or not. By the symmetry of p and q, we have the following three possibilities to consider:
   (a) $\{p, q\} \subseteq W$,
   (b) $p \in W$, $q \notin W$,
   (c) $\{p, q\} \cap W = \emptyset$.

We analyze these cases in order.

(a) $\{p, q\} \subseteq W$: This is the easiest case. Since by assumption $r \in B$ and $\{p, q\} \subseteq W$, the subconfiguration $[B]\langle W\rangle \in \mathbb{R}(\mathbb{C}_t)$ can be derived by a download of $[r]\langle p, q\rangle$ followed by a weakening of $[r]\langle p, q\rangle$ to $[B \cup r]\langle W \cup \{p, q\}\rangle = [B]\langle W\rangle$.

(b) $p \in W$, $q \notin W$: In this case $[r]\langle p, q\rangle$ cannot be weakened to $[B]\langle W\rangle$, since $q \notin W$. We need to find some way to eliminate the white pebble on q. But since $q \notin W$, part 2 of Observation 10.15 says that we can derive $[B \cup q]\langle W\rangle$ by weakening from $\mathbb{R}(\mathbb{C}_{t-1})$. Using this subconfiguration, we can derive $[B]\langle W\rangle$ as follows:

- download $[r]\langle p, q\rangle$,
- derive $[B \cup q]\langle W\rangle$ from $\mathbb{R}(\mathbb{C}_{t-1})$ by weakening,
- resolve $[r]\langle p, q\rangle$ and $[B \cup q]\langle W\rangle$ to get $[B \cup r]\langle W \cup p\rangle = [B]\langle W\rangle$.

Note that the resolution move is in accordance with the rules since $\{r\} \cap W = \emptyset$ as noted in (10.25) and $(B \cup \{q\}) \cap \{p, q\} = \{q\}$ as noted in (10.24).

(c) $\{p, q\} \cap W = \emptyset$: Now both p and q have to be eliminated if we are to use $[r]\langle p, q\rangle$ to derive $[B]\langle W\rangle$, but by applying part 2 of Observation 10.15 twice we see that we can derive $[B \cup p]\langle W\rangle$ and $[B \cup q]\langle W\rangle$ by weakening from $\mathbb{R}(\mathbb{C}_{t-1})$. Using this fact, we can perform a pebbling to get $[B]\langle W\rangle$ as follows:

- download $[r]\langle p, q\rangle$,
- derive $[B \cup q]\langle W\rangle$ from $\mathbb{R}(\mathbb{C}_{t-1})$ by weakening,
- resolve $[r]\langle p, q\rangle$ with $[B \cup q]\langle W\rangle$ on q to get $[B \cup r]\langle W \cup p\rangle = [B]\langle W \cup p\rangle$,
- derive $[B \cup p]\langle W\rangle$ from $\mathbb{R}(\mathbb{C}_{t-1})$ by weakening,
- conclude by resolving $[B]\langle W \cup p\rangle$ with $[B \cup p]\langle W\rangle$ on p, resulting in the subconfiguration $[B]\langle W\rangle$.

Of course, it needs to be checked that all resolution moves are legal, but this follows from (10.24) and (10.25).

This concludes the analysis for the case $r \in B$ for a non-source vertex r.

2. $r \notin B$: This case is quite similar to the previous case $r \in B$. Here also we make a subcase analysis depending on whether $|\mathit{pred}(r) \cap W|$ is equal to 2, 1 or 0.

Before we do this, though, we observe that there is a particular subconfiguration that will be useful for us. Since we are now assuming that $r \notin B$, part 1 of Observation 10.15 says that $[B]\langle W \cup r\rangle$ is derivable by weakening from $\mathbb{R}(\mathbb{C}_{t-1})$. This subconfiguration will play an important role in the pebblings below.

(a) $\{p, q\} \subseteq W$: To get the subconfiguration $[B]\langle W\rangle$ from $\mathbb{R}(\mathbb{C}_{t-1})$ in this case, first derive the subconfiguration $[B]\langle W \cup r\rangle$ just mentioned by weakening from $\mathbb{R}(\mathbb{C}_{t-1})$, then download $[r]\langle p, q\rangle$, and finally resolve the two to get $[B]\langle W \cup \{p, q\}\rangle = [B]\langle W\rangle$. This resolution move is in accordance with the rules since $B \cap \{p, q\} = \emptyset$ according to (10.24) and $\{r\} \cap (W \cup \{r\}) = \{r\}$.

(b) $p \in W$, $q \notin W$: Just as in case 1b, part 2 of Observation 10.15 says that $[B \cup q]\langle W\rangle$ is derivable from $\mathbb{R}(\mathbb{C}_{t-1})$ by weakening. Now do the following pebbling moves:

- download $[r]\langle p, q\rangle$,
- derive $[B \cup q]\langle W\rangle$ from $\mathbb{R}(\mathbb{C}_{t-1})$ by weakening using part 2 of Observation 10.15 as in case 1b,
- resolve $[r]\langle p, q\rangle$ with $[B \cup q]\langle W\rangle$ on q to get $[B \cup r]\langle W \cup p\rangle$,
- use part 1 of Observation 10.15 to derive $[B]\langle W \cup r\rangle$ by weakening, from $\mathbb{R}(\mathbb{C}_{t-1})$ by weakening,
- finally, resolve $[B \cup r]\langle W \cup p\rangle$ with $[B]\langle W \cup r\rangle$ on the vertex r to get $[B]\langle W \cup p\rangle = [B]\langle W\rangle$.

(c) $\{p, q\} \cap W = \emptyset$: As in case 1c, appeal to part 2 of Observation 10.15 twice to find subconfigurations $[B \cup p]\langle W\rangle, [B \cup q]\langle W\rangle$ derivable from $\mathbb{R}(\mathbb{C}_{t-1})$ by weakening. Using that $[B]\langle W \cup r\rangle$ also can be derived from $\mathbb{R}(\mathbb{C}_{t-1})$ by weakening, we can make the following sequence of pebbling moves:

- download $[r]\langle p, q\rangle$,
- derive $[B \cup q]\langle W\rangle$ by weakening,
- resolve $[r]\langle p, q\rangle$ and $[B \cup q]\langle W\rangle$ on q to derive $[B \cup r]\langle W \cup p\rangle$,
- derive $[B \cup p]\langle W\rangle$ by weakening,
- resolve $[B \cup r]\langle W \cup p\rangle$ and $[B \cup p]\langle W\rangle$ on p to derive $[B \cup r]\langle W\rangle$,
- derive $[B]\langle W \cup r\rangle$ by weakening,
- finally, resolve $[B \cup r]\langle W\rangle$ and $[B]\langle W \cup r\rangle$ on r resulting in $[B]\langle W\rangle$.

Double-checking the set intersections and inclusions shows that all these moves are legal.

This concludes the analysis for the case $r \notin B$.

10.4.6 Summing up the Proof of Theorem 10.13

If $\pi = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ is a derivation of $Cl[f_d(\vec{z})]$ from $^*Peb^d_G[f]$, it is easily verified from Definition 10.12 that $\mathbb{R}(\mathbb{C}_0) = \mathbb{R}(\emptyset) = \emptyset$ and $\mathbb{R}(\mathbb{C}_\tau) = \mathbb{R}(Cl[f_d(\vec{z})]) = \{[z]\langle\emptyset\rangle\}$.

In Sections 10.4.3, 10.4.4, and 10.4.5, we have shown how to do the intermediate res-pebbling moves to get from $\mathbb{S}(\mathbb{C}_{t-1})$ to $\mathbb{S}(\mathbb{C}_t)$ in the case of erasure, inference and axiom download, respectively. For erasure and inference, the blob-pebbling cost changes monotonically during the transition $\mathbb{S}(\mathbb{C}_{t-1}) \rightsquigarrow \mathbb{S}(\mathbb{C}_t)$. In the case of axiom download, all pebbles used in the intermediate moves are still on the DAG in $\mathbb{S}(\mathbb{C}_t)$ except possibly for the pebbles on $\{r\} \cup \mathit{pred}(r)$, so the extra intermediate cost is upper-bounded by 3.

This shows that the complete res-pebbling $\mathcal{R}_\pi$ of the DAG G associated to any resolution derivation $\pi : {}^*\!\mathit{Peb}^d_G[f] \vdash \mathit{Cl}[f_d(\vec{z})]$ by the construction in this section has res-pebbling cost bounded from above by $\mathsf{cost}(\mathcal{R}_\pi) \leq \max_{\mathbb{C}\in\pi}\bigl\{\mathsf{cost}(\mathbb{R}(\mathbb{C}))\bigr\} + 3$. Theorem 10.13 is thereby proven.

10.5 Res-Pebbling Price Lower-Bounds Space

Everything said so far, and, in particular, everything in Section 10.4, applies to any (non-constant) Boolean function f and any pebbling degree $d \in \mathbb{N}^+$. In order to clinch the proof of our lower bounds on space, however, in this final section we need to focus on a particular kind of functions $f(x_1, \ldots, x_d)$ having the property that no single variable x_i determines the value of $f(x_1, \ldots, x_d)$. In particular, this means that we need at least $d \geq 2$ variables per vertex.

Definition 10.16 (Non-authoritarian function). We will call a Boolean function f over d variables $x_1, \ldots, x_d$ *non-authoritarian* if for any variable x_i and any truth value $\alpha(x_i) = \nu_i$ assigned to x_i, α can be extended to a truth value assignment α' satisfying $f(x_1, \ldots, x_d)$ and another truth value assignment α'' falsifying $f(x_1, \ldots, x_d)$. If f does not satisfy this requirement, then we will call the function *authoritarian*.

The following observations are immediate:

- The functions $\bigoplus_{i=1}^d x_i$ are non-authoritarian if $d > 1$.
- The functions $k\text{-}\mathit{true}_d(x_1, \ldots, x_d)$ are non-authoritarian if $1 < k < d$.
- However, all functions $\bigvee_{i=1}^d x_i = 1\text{-}\mathit{true}_d(x_1, \ldots, x_d)$ are authoritarian.

As opposed to the analogous results in Chapters 8 and 9, which were proven for the authoritarian function $\bigvee_{i=1}^d x_i$, the lower bounds proven in this section will depend crucially on the fact that the function $f(x_1, \ldots, x_d)$ that we use is non-authoritarian. What we are going to prove is that if a set of clauses $\mathbb{C}$ derived from $\mathit{Peb}^d_G[f]$ for some non-authoritarian function f induces a res-pebbling configuration $\mathbb{R}(\mathbb{C})$ according to Definition 10.12, then the cost of $\mathbb{R}(\mathbb{C})$ as specified in Definition 10.8 is at most $|\mathbb{C}|$. That is, the cost of an induced res-pebbling configuration provides a lower bound on the size of the set of clauses inducing it.

Recall that we say that a vertex u is represented in a clause C, and that C mentions u, if $\mathit{Vars}^d(u) \cap \mathit{Vars}(C) \neq \emptyset$, where $\mathit{Vars}^d(u) = \{u_1, \ldots, u_d\}$. We write $V(\mathbb{C})$ to denote all vertices represented in a clause set $\mathbb{C}$.

The following lemma says that if $[B]\langle W\rangle$ is induced by $\mathbb{C}$, then all vertices in $B \cup W$ are represented in any minimal subset of clauses $\mathbb{C}' \subset \mathbb{C}$ inducing $[B]\langle W\rangle$. This holds for any function f.

Lemma 10.17. *Consider the pebbling contradiction* $^*Peb^d_G[f]$ *for any non-constant Boolean function* f *and suppose that* $\mathbb{C}$ *is a set of clauses derived from* $^*Peb^d_G[f]$ *such that* $[B]\langle W\rangle \in \mathbb{R}(\mathbb{C})$*. Then* $B \cup W \subseteq V(\mathbb{C})$*.*

Proof. Fix any subset of clauses $\mathbb{C}' \subseteq \mathbb{C}$ such that $\mathbb{C}' \rhd \bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w})$. Consider any $v \in (B \cup W)\setminus V(\mathbb{C}')$ and suppose without loss of generality that $v \in B$. By Definition 10.12 it holds that $\mathbb{C}' \nvDash \bigvee_{b\in B\setminus\{v\}} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w})$, so there is an assignment α such that $\alpha(\mathbb{C}') = 1$ and $\alpha\big(\bigvee_{b\in B\setminus\{v\}} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w})\big) = 0$. Change the values of the variables $v_1, \ldots, v_d$ in α as needed to get an α' with $\alpha'(f(v_1, \ldots, v_d)) = 0$. Then $\alpha'\big(\bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w})\big) = 0$, since we did not touch $\mathit{Vars}^d\big((B \cup W) \setminus \{v\}\big)$, so we must have $\alpha'(\mathbb{C}') = 0$. But this means that there is some clause $C \in \mathbb{C}'$ with $\alpha(C) = 1$ but $\alpha'(C) = 0$, and this clause mentions v. □

Using Lemma 10.17 and in addition assuming that f is non-authoritarian, we can prove the lower bound that we need.

Theorem 10.18. *Suppose that* $\mathbb{C}$ *is a set of clauses derived from the pebbling contradiction* $^*Peb^d_G[f]$ *for any non-authoritarian function* f*. Then* $|\mathbb{C}| > \mathsf{cost}(\mathbb{R}(\mathbb{C})) = \big|\bigcup_{[B]\langle W\rangle\in\mathbb{R}(\mathbb{C})}(B \cup W)\big|$*.*

Proof. Let us write

$$V^* = \bigcup_{[B]\langle W\rangle\in\mathbb{R}(\mathbb{C})}(B \cup W) \tag{10.26}$$

to denote all pebbled vertices in the configuration induced by $\mathbb{C}$. Consider the bipartite graph with clauses in $\mathbb{C}$ on the left-hand side and vertices in V^* on the right-hand side. We draw an edge between $C \in \mathbb{C}$ and $v \in V^*$ if C mentions v. That is, the set of neighbours of C is $N(C) = V(C) \cap V^*$. By Lemma 10.17 we have $V^* \subseteq V(C)$, so every $v \in V^*$ is the neighbour of some $C \in \mathbb{C}$.

Let $\mathbb{C}_1 \subseteq \mathbb{C}$ be a set of maximal size such that $|\mathbb{C}_1| > |N(\mathbb{C}_1)|$. If $\mathbb{C}_1 = \mathbb{C}$ we are done, so suppose this is not the case. We show that $\mathbb{C}_1 \neq \mathbb{C}$ leads to a contradiction.

To this end, let $\mathbb{C}_2 = \mathbb{C}\setminus\mathbb{C}_1 \neq \emptyset$ and define the vertex sets $V_1^* = N(\mathbb{C}_1)$ and $V_2^* = V^* \setminus V_1^*$. Note that we must have $V_2^* \subseteq N(\mathbb{C}_2)$ since $N(\mathbb{C}) = N(\mathbb{C}_1) \cup N(\mathbb{C}_2) = V^*$. By the maximality of $\mathbb{C}_1$ it must hold for all $\mathbb{D} \subseteq \mathbb{C}_2$ that $|\mathbb{D}| \leq \big|N(\mathbb{D})\setminus V_1^*\big|$, because otherwise $\mathbb{C}' = \mathbb{C}_1 \cup \mathbb{D}$ would be a larger set with $|\mathbb{C}'| > |N(\mathbb{C}')|$. But this means that by Hall's marriage theorem, there is a matching M of $\mathbb{C}_2$ into $N(\mathbb{C}_2)\setminus V_1^* = V_2^*$. Consider any subconfiguration $[B]\langle W\rangle$ such that $(B \cup W) \cap V_2^* \neq \emptyset$ and let $\mathbb{C}' \subseteq \mathbb{C}$ be any clause set such that

$$\mathbb{C}' \rhd \bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w}) \ . \tag{10.27}$$

We claim that we can construct a truth value assignment α that makes $\mathbb{C}'$ true but $\bigvee_{b\in B} f_d(\vec{b}) \vee \bigvee_{w\in W} \neg f_d(\vec{w})$ false. This is clearly a contradiction, and so the theorem follows.

To prove the claim, let $\mathbb{C}'_i = \mathbb{C}' \cap \mathbb{C}_i$, $B_i = B \cap V_i^*$, and $W_i = W \cap V_i^*$ for $i = 1, 2$. Since $B_1 \cup W_1 \subsetneqq B \cup W$ by construction, the minimality condition in (10.27) yields that

$$\mathbb{C}'_1 \nvDash \bigvee_{b \in B_1} f_d(\vec{b}) \vee \bigvee_{w \in W_1} \neg f_d(\vec{w}) \tag{10.28}$$

so we can find a truth value assignment α_1 that sets $\mathbb{C}'_1$ to true, all $f(b_1, \ldots, b_d)$, $b \in B_1$, to false, and all $f(w_1, \ldots, w_d)$, $w \in W_1$, to true. Note that α_1 need only assign values to variables in $\mathit{Vars}^d\big(B_1 \cup W_1\big)$. For $\mathbb{C}'_2$, we use the matching M into V_2^* to find a distinct vertex $v(C)$ for every $C \in \mathbb{C}'_2$ and a literal over some variable $v(C)_i \in \mathit{Vars}^d\big(v(C)\big)$ that fixes C to true. Let α_2 be this assignment. We stress that α_2 assign values to at most one variable v_i for any $v \in B_2 \cup W_2$. But since f is non-authoritarian, this means that we can extend α_2 to an assignment still satisfying $\mathbb{C}'_2$ but setting all $f(b_1, \ldots, b_d)$, $b \in B_2$, to false and all $f(w_1, \ldots, w_d)$, $w \in W_2$, to true. It follows that $\alpha = \alpha_1 \cup \alpha_2$ is a truth value assignment such that $\alpha(\mathbb{C}') = 1$ but $\alpha(\bigvee_{b \in B} f_d(\vec{b}) \vee \bigvee_{w \in W} \neg f_d(\vec{w})) = 0$, which proves the claim. □

Let us conclude this section by recollecting why Theorem 2.5 and Corollary 2.6 now follow.

Theorem 10.19 (strengthened version of Theorem 2.5). *Let $\mathit{Peb}^d_G[f]$ be a pebbling contradiction for any single-sink fan-in 2 DAG G, any fixed pebbling degree $d \geq 2$, and any non-authoritarian function $f(x_1, \ldots, x_d)$. Then it holds that $\mathit{Sp}\big(\mathit{Peb}^d_G[f] \vdash 0\big) \geq \textsf{BW-Peb}(G) - \mathrm{O}(1)$.*

Proof. Consider instead the pebbling formula $^*\mathit{Peb}^d_G[f]$ without sink axioms. By Lemma 10.11 it holds that $\mathit{Sp}\big(\mathit{Peb}^d_G[f] \vdash 0\big) = \mathit{Sp}\big(^*\mathit{Peb}^d_G[f] \vdash \mathit{Cl}[f_d(\vec{z})]\big) + \mathrm{O}(1)$ if d is constant, so it is sufficient to prove lower bounds on the clause space of deriving $\mathit{Cl}[f_d(\vec{z})]$ from $^*\mathit{Peb}^d_G[f]$.

Fix any resolution derivation $\pi : {}^*\mathit{Peb}^d_G[f] \vdash \mathit{Cl}[f_d(\vec{z})]$ and let $\mathcal{R}_\pi$ be the complete res-pebbling of the graph G associated to π in Theorem 10.13 such that $\textsf{cost}(\mathcal{R}_\pi) \leq \max_{\mathbb{C} \in \pi}\big\{\textsf{cost}(\mathbb{R}(\mathbb{C}))\big\} + \mathrm{O}(1)$. On the one hand, Theorem 10.18 says that $\textsf{cost}(\mathbb{R}(\mathbb{C})) \leq |\mathbb{C}|$ for any $\mathbb{C}$ since f is non-authoritarian, so, in particular, it must hold that $\textsf{cost}(\mathcal{R}_\pi) \leq \mathit{Sp}(\pi) + \mathrm{O}(1)$. On the other hand, $\textsf{cost}(\mathcal{R}_\pi) \geq \textsf{Res-Peb}(G) \geq \textsf{BW-Peb}(G)$ by Theorem 10.3. Thus $\mathit{Sp}(\pi) \geq \textsf{BW-Peb}(G) - \mathrm{O}(1)$, and the theorem follows. To get a statement of the form of Theorem 2.5, choose f to be the non-authoritarian function $f(x_1, x_2) = x_1 \oplus x_2$. □

Proving Corollary 2.6 is now just a matter of picking the right DAGs G_n and the right function f.

Corollary 2.6 (restated). *For all $k \geq 6$ there is a family $\{F_n\}_{n=1}^{\infty}$ of k-CNF formulas of size $\mathrm{O}(n)$ such that $L(F_n \vdash 0) = \mathrm{O}(n)$ and $W(F_n \vdash 0) = \mathrm{O}(1)$ but $\mathit{Sp}(F_n \vdash 0) = \Omega(n/\log n)$.*

Proof. Let $\{G_n\}_{n=1}^{\infty}$ be the family of single-sink fan-in 2 DAGs in Theorem 5.5 with size $\mathrm{O}(n)$ and black-white pebbling price $\mathit{BW\text{-}Peb}(G_n) = \Omega(n/\log n)$. Then for any non-constant Boolean function f, the pebbling contradiction $\mathit{Peb}^d_{G_n}[f]$ can be refuted in length $L\bigl(\mathit{Peb}^d_{G_n}[f] \vdash 0\bigr) = \mathrm{O}(n)$ and width $W\bigl(\mathit{Peb}^d_{G_n}[f] \vdash 0\bigr) = \mathrm{O}(1)$ if d is fixed to some constant by Proposition 10.5.

To get a lower bound on the clause space, we just pick f to be some suitable non-authoritarian function. As has already been noted, functions $\bigoplus_{i=1}^{d} x_i$ work fine, but it would be nice to get the corollary for any clause width $w \geq 6$ and not only for clause widths divisible by 3. Therefore, we instead consider pebbling contradictions $\mathit{Peb}^d_G[k\text{-}\mathit{true}]$. More precisely, for any fixed width w we choose f to be the function $2\text{-}\mathit{true}_d(x_1, \ldots, x_d)$ for $d = w - 3$ specifying that $f_d(x_1, \ldots, x_d)$ is true precisely when there are at least 2 distinct variables x_i and x_j being true. Since $1 < 2 < d$ if $w \geq 6$, $2\text{-}\mathit{true}_d(x_1, \ldots, x_d)$ is non-authoritarian, and as was noted above the width of $\mathit{Peb}^d_G[k\text{-}\mathit{true}]$ is $d + k + 1 = w$. Hence, the corollary follows by applying Theorem 2.5 on 2-true-pebbling contradictions. □

We also note that somewhat intriguingly, width $k = 6$ is actually a tight lower bound for the techniques used in this chapter (as opposed to in Chapters 8 and 9, where the corresponding weaker bound on space can be made to hold for k-CNF formulas of any width $k \geq 4$). This is a consequence of the following easy observation.

Observation 10.20. *If f_d is a non-authoritarian function, then the clause width of the corresponding pebbling contradiction $\mathit{Peb}^d_G[f]$ is at least* 6*.*

Proof. If $W\bigl(\mathit{Cl}[f_d(\vec{x})]\bigr) = 1$ or $W\bigl(\mathit{Cl}[\neg f_d(\vec{x})]\bigr) = 1$, there would exist a single variables x_i being able to determine the value of f. Thus, we must have $W\bigl(\mathit{Peb}^d_G[f]\bigr) = 2 \cdot W(\mathit{Cl}[\neg f_d(\vec{x})]) + W(\mathit{Cl}[f_d(\vec{x})]) \geq 6$. □

Chapter 11

Understanding Space in Resolution

This chapter contains a brief report on ongoing work with Eli Ben-Sasson generalizing the results in Chapter 10. It would be interesting to see if one could gain additional insights from recasting Chapters 8 and 9 in this framework as well, but we have not had the time to study that question.

We want to stress that the results presented in this chapter have been obtained very, very recently, even more so than the results in Chapter 10, and the material below is all very much work in progress at the time of submission of this thesis.

11.1 Substitution Formulas and Variable Support Size

Throughout this chapter, we will let f_d denote any non-constant Boolean function $f_d : \{0,1\}^d \mapsto \{0,1\}$ of arity d. We use the shorthand $\vec{x} = (x_1, \ldots, x_d)$, so that $f_d(\vec{x})$ is just an equivalent way of writing $f_d(x_1, \ldots, x_d)$.

As in Chapter 10, we assume some fixed canonical representations of f_d and $\neg f_d$ as CNF formulas and write $\mathit{Cl}[f_d(\vec{x})]$ and $\mathit{Cl}[\neg f_d(\vec{x})]$ to denote the CNF formulas over $x_1, \ldots, x_d$ that are logically equivalent to $f_d(x_1, \ldots, x_d)$ and $\neg f_d(x_1, \ldots, x_d)$, respectively. Some examples of this notation were given in Section 10.2.

We want to define formally what it means to substitute f_d for the variables $\mathit{Vars}(F)$ in a CNF formula F. For notational convenience, we assume that F only has variables x, y, z, et cetera, without subscripts, so that $x_1, \ldots, x_d$, $y_1, \ldots, y_d$, $z_1, \ldots, z_d, \ldots$ are new variables not occurring in F.

Definition 11.1 (Substitution formula). For a positive literal x, we define the *f_d-substitution* of x to be $x[f_d] = \mathit{Cl}[f_d(\vec{x})]$, i.e., the canonical representation of $f_d(x_1, \ldots, x_d)$ as a CNF formula. For a negative literal $\overline{y}$, the f_d-substitution is $\overline{y}[f_d] = \mathit{Cl}[\neg f_d(\vec{y})]$. The f_d-substitution of a clause $C = a_1 \lor \cdots \lor a_k$ is the CNF formula

$$C[f_d] = \bigwedge_{C_1 \in a_1[f_d]} \cdots \bigwedge_{C_k \in a_k[f_d]} \bigl(C_1 \lor \ldots \lor C_k\bigr) \qquad (11.1)$$

and the f_d-substitution of a CNF formula F is $F[f_d] = \bigwedge_{C \in F} C[f_d]$.

With this notation, the generalized pebbling contradictions $\mathit{Peb}^{d}_{G}[f_d]$ in Chapter 10 are f_d-substitutions of first-degree pebbling contradictions Peb^{1}_{G}, and hence Definition 11.1 is just a natural generalization of Definition 10.4. As another, more concrete example, for $C = x \vee \overline{y}$ and $f_2 = x_1 \oplus x_2$ we get that

$$\begin{aligned} C[f_2] = \quad & (x_1 \vee x_2 \vee y_1 \vee \overline{y}_2) \wedge (x_1 \vee x_2 \vee \overline{y}_1 \vee y_2) \\ \wedge\ & (\overline{x}_1 \vee \overline{x}_2 \vee y_1 \vee \overline{y}_2) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{y}_1 \vee y_2) \ . \end{aligned} \tag{11.2}$$

We note that $F[f_d]$ is a CNF formula over $d \cdot |\mathit{Vars}(F)|$ variables containing at most $|F| \cdot 2^{d \cdot W(F)}$ clauses. (Recall that we defined a CNF formula as a set of clauses, which means that $|F|$ is the number of clauses in F.)

We have the following easy observation, the proof of which is presented for completeness.

Observation 11.2. *For any non-constant Boolean function $f_d : \{0,1\}^d \mapsto \{0,1\}$, it holds that $F[f_d]$ is unsatisfiable if and only if F is unsatisfiable.*

Proof. Suppose that F is satisfiable and let α be a truth value assignment such that $\alpha(F) = 1$. Then we can satisfy $F[f_d]$ by choosing a truth value assignment α' for $\mathit{Vars}\bigl(F[f_d]\bigr)$ in such a way that $f_d\bigl(\alpha'(x_1), \ldots, \alpha'(x_d)\bigr) = \alpha(x)$. For if $C \in F$ is satisfied by some literal a_i set to true by α, then α' will satisfy all clauses $C_i \in a_i[f_d]$ and thus also the whole CNF formula $C[f_d]$ in (11.1).

Conversely, suppose F is unsatisfiable and consider any truth value assignment α' for $F[f_d]$. Then α' defines a truth value assignment α for F in the natural way by setting $\alpha(x) = f_d\bigl(\alpha'(x_1), \ldots, \alpha'(x_d)\bigr)$, and we know that there is some clause $C \in F$ that is not satisfied by α. That is, for every literal $a_i \in C = a_1 \vee \cdots \vee a_k$ it holds that $\alpha(a_i) = 0$. But then α' does not satisfy $a_i[f_d]$, so there is some clause $C'_i \in a_i[f_d]$ such that $\alpha'(C'_i) = 0$. This shows that α' falsifies the disjunction $C'_1 \vee \cdots \vee C'_k \in C[f_d]$, and consequently $F[f_d]$ must also be unsatisfiable. □

To state the results in this chapter, we also need to define a new proof complexity measure which is related to, but weaker than, variable space.

Definition 11.3 (Variable support size). Let us say that the *variable support size*, or just *support size*, of a clause set $\mathbb{C}$ is $\mathit{SuppSize}(\mathbb{C}) = |\mathit{Vars}(\mathbb{C})|$, i.e., the number of variables mentioned in $\mathbb{C}$. We define the support size of a resolution derivation $\pi = \{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ to be $\mathit{SuppSize}(\pi) = \max_{t \in [\tau]}\{\mathit{SuppSize}(\mathbb{C})\}$ and the minimal support size of refuting F is then $\mathit{SuppSize}(F \vdash 0) = \min_{\pi : F \vdash 0}\{\mathit{SuppSize}(\pi)\}$.

The difference between variable space and variable support size is that the variables space counts the number of variable occurrences in $\mathbb{C}$ *with repetitions*, but for variable support size we only count each variable once no matter how often it occurs. It follows that the support size of refuting a formula is always at most linear in the formula size, while the refutation variable space could potentially be quadratic in the formula size in the worst case. (As was noted in Section 4.3, though, no such formulas are known to exist, and we do not even know of any superlinear lower bounds on variable space.)

Although the next result is stated for variable space in [16] (as quoted in Theorem 5.11), the proof in [16] actually establishes the slightly stronger statement below.

Theorem 11.4 ([16]). *For any DAG G, it holds that $\mathit{SuppSize}(\mathit{Peb}_G^1 \vdash 0) \geq \mathsf{BW\text{-}Peb}(G)$.*

Combining Theorem 11.4 with Theorem 5.5 and Proposition 5.7, we get the following corollary.

Corollary 11.5 ([16]). *There exists a family of 3-CNF formulas $\{F_n\}_{n=1}^{\infty}$ with $\mathrm{O}(n)$ clauses and variables such that there are resolution refutations $\pi_n : F_n \vdash 0$ in length $L(\pi_n) = \mathrm{O}(n)$ and width $W(\pi_n) = \mathrm{O}(1)$, but the support size of any refutation is $\mathit{SuppSize}(F_n \vdash 0) = \Omega(n/\log n)$.*

11.2 A General Theorem on Substitution and Space

Recall from Definition 10.16 that a non-authoritarian function f_d is a function such that no single variable x_i can determine the value of $f_d(x_1, \ldots, x_d)$. Our next theorem says that we can convert lower bounds on variable support size into lower bounds on clause space by making f_d-substitutions using non-authoritarian functions. Thus, the innocent-looking Corollary 11.5 above is in fact all we need to get an optimal separation of clause space and length in resolution.

Theorem 11.6. *Let F be any unsatisfiable CNF formula and $f_d : \{0,1\}^d \mapsto \{0,1\}$ be any non-constant Boolean function. Then it holds that the substitution formula $F[f_d]$ can be refuted in width*

$$W\bigl(F[f_d] \vdash 0\bigr) = \mathrm{O}\bigl(d \cdot W(F \vdash 0)\bigr)$$

and length

$$L\bigl(F[f_d] \vdash 0\bigr) \leq \min_{\pi : F \vdash 0} \bigl\{L(\pi) \cdot \exp\bigl(\mathrm{O}(d \cdot W(\pi))\bigr)\bigr\} \ .$$

If in addition f_d is non-authoritarian, it holds that

$$\mathit{Sp}\bigl(F[f_d] \vdash 0\bigr) \geq \mathit{SuppSize}(F \vdash 0) \ ,$$

i.e., the clause space of refuting the substitution formula $F[f_d]$ is lower-bounded by the variable support size of refuting the original formula F.

Note that if F is refutable simultaneously in linear length and constant width, then the bound in Theorem 11.6 on $L\bigl(F[f_d] \vdash 0\bigr)$ becomes linear in $L(F \vdash 0)$. And the goal of the long, hard work in the papers [60, 63, 20] can be obtained instead as an easy corollary of this theorem.

Corollary 11.7 (Corollary 2.6 restated). *There is a family* $\{F_n\}_{n=1}^{\infty}$ *of* k*-CNF formulas of size* $\mathrm{O}(n)$ *such that* $L(F_n \vdash 0) = \mathrm{O}(n)$ *and* $W(F_n \vdash 0) = \mathrm{O}(1)$ *but* $\mathit{Sp}(F_n \vdash 0) = \Omega(n/\log n)$.

Proof. Fix any non-authoritarian function f_d, do f_d-substitution on the formulas in Corollary 11.5, and appeal to Theorem 11.6. □

11.3 Proof Sketch for the Substitution Space Theorem

Due to the fact that Theorem 11.6 was discovered only at a very late stage when this thesis was already supposed to be ready to go to press, time constraints do not allow us to present a full, polished proof. In this section, however, we try to provide enough details to convince the reader that the theorem is indeed true.

The upper bounds on refutation width and length for $F[f_d]$ are not hard. Given a resolution refutation π of F, we construct a refutation $\pi' : F[f_d] \vdash 0$ mimicking the derivation steps in π. When π downloads an axiom C, we download the $\exp\bigl(\mathrm{O}(d \cdot W(C))\bigr)$ axiom clauses in $C[f_d]$. When π resolves $C_1 \lor x$ and $C_2 \lor \overline{x}$ to derive $C_1 \lor C_2$, we use the fact that resolution is implicationally complete to derive $(C_1 \lor C_2)[f_d]$ from $(C_1 \lor x)[f_d]$ and $(C_2 \lor \overline{x})[f_d]$.

The interesting part of the theorem, however, is of course how we can get lower bounds on clause space for $F[f_d]$ from lower bounds on support size for F. The idea is to look at refutations of $F[f_d]$ and "project" them down on refutations of F. To do this, we re-use the definition of precise implication from Definitions 9.9 and 10.12.

Let us also, in this section, use the convention that any clause C can be written $C = C^+ \lor C^-$, where $C^+ = \bigvee_{x \in \mathit{Lit}(C)} x$ is the disjunction of the positive literals in C and $C^- = \bigvee_{\overline{y} \in \mathit{Lit}(C)} \overline{y}$ is the disjunction of the negative literals. Also, for brevity we do not write $x \in \mathit{Lit}\bigl(C^+\bigr)$ or $\overline{y} \in \mathit{Lit}\bigl(C^-\bigr)$ below, but instead $x \in C^+$ and $\overline{y} \in C^-$ (which is still formally correct since a clause is a set of literals).

Definition 11.8 (Projected clauses). Let F be a CNF formula and f_d a non-constant Boolean function, and suppose that $\mathbb{D}$ is a set of clauses derived from $F[f_d]$. Then we say that $\mathbb{D}$ *projects* the clause $C = C^+ \lor C^-$ on F (or, perhaps more correctly, on $\mathit{Vars}(F)$) if there is a subset of clauses $\mathbb{D}_C \subseteq \mathbb{D}$ such that

$$\mathbb{D}_C \rhd \bigvee_{x \in C^+} f_d(\vec{x}) \lor \bigvee_{\overline{y} \in C^-} \neg f_d(\vec{y}) \tag{11.3}$$

and we write

$$\mathit{proj}_F(\mathbb{D}) = \bigl\{ C \bigm| \exists\, \mathbb{D}_C \subseteq \mathbb{D} \text{ s.t. } \mathbb{D}_C \rhd \textstyle\bigvee_{x \in C^+} f_d(\vec{x}) \lor \bigvee_{\overline{y} \in C^-} \neg f_d(\vec{y}) \bigr\} \tag{11.4}$$

to denote the set of all clauses that $\mathbb{D}$ projects on F.

Given that we now know how to translate clauses derived from $F[f_d]$ into clauses over $\mathit{Vars}(F)$, the next step is to show that this translation preserves resolution refutations.

Theorem 11.9. *Suppose that $\pi = \{\mathbb{D}_0, \ldots, \mathbb{D}_\tau\}$ is a resolution refutation of $F[f_d]$ for some arbitrary unsatisfiable CNF formula F and some arbitrary non-constant function f_d. Then the sets of projected clauses $\{\mathit{proj}_F(\mathbb{D}_0), \ldots, \mathit{proj}_F(\mathbb{D}_\tau)\}$ form the "backbone" of a resolution refutation π_F of F in the sense that $\mathit{proj}_F(\mathbb{D}_0) = \emptyset$, $\mathit{proj}_F(\mathbb{D}_\tau) = \{0\}$, and all transitions from $\mathit{proj}_F(\mathbb{D}_{t-1})$ to $\mathit{proj}_F(\mathbb{D}_t)$ for $t \in [\tau]$ can be accomplished by axiom downloads from F, inferences, erasures, and possibly weakening steps in such a way that the variable support size in π_F during these intermediate derivation steps never exceeds $\max_{\mathbb{D} \in \pi}\{\mathit{SuppSize}(\mathit{proj}_F(\mathbb{D}))\}$.*

Section 10.4 does in fact present a proof of this theorem for the special case when $F = \mathit{Peb}^1_G$. The full proof of the general case is obtained by going through the construction in Section 10.4 carefully and see that it can in fact be made to work for arbitrary CNF formulas (and that additionally, we can get rid of the O(1) term in the final bound).

If we are ready to believe that Theorem 11.9 can be proven in this way, the rest is now easy. We just need the following fact.

Theorem 11.10. *Suppose that $\mathbb{D}$ is a non-contradictory set of clauses derived from $F[f_d]$ for some arbitrary unsatisfiable CNF formula F and some non-authoritarian function f_d. Then $\mathit{Sp}(\mathbb{D}) \geq \mathit{SuppSize}(\mathit{proj}_F(\mathbb{D}))$.*

But this is exactly Theorem 10.18, albeit expressed in different notation, so we can just copy the proof in Section 10.5. And combining Theorems 11.9 and 11.10, Theorem 11.6 follows. This concludes our proof sketch.

Part IV

Conclusion

Chapter 12

Concluding Remarks

The main contribution of this thesis is the sequence of results [60, 63, 20] leading to an optimal separation of clause space and length in resolution. The same results also yield an almost optimal separation of clause space and width. This answers two questions in proof complexity that have been open for some years.

Furthermore, as a part of the ongoing research reported in Chapter 11 we observe that our separation results appear to be an immediate corollary of a more general (and more simple) statement about how properties of CNF formulas change under substitution. We hope that there should be further interesting implications of this theorem and are currently working on exploring this issue.

We can also see several other natural and interesting follow-up questions to our results. This final chapter is an attempt to briefly outline some of these questions. While Section 12.1 mentions some remaining issues from our previous papers that are raised mostly out of pure curiosity, Sections 12.2 and 12.3 discuss more fundamental problems that seem interesting to pursue in the future.

12.1 Resolution Refutation Space of Pebbling Contradictions

The lower bounds on clause space in Chapters 8 and 9 were proven for the "standard" pebbling contradictions $Peb^d_G[\vee]$ in Definition 5.6. In order to get the optimal separation result in Chapter 10, however, we had to switch formulas to more general pebbling contradictions $Peb^d_G[f]$ in Definition 10.4 for other functions f than logical or. So, strictly speaking, the open question in, for instance, [16] about the refutation clause space of pebbling contradictions remains.

Open Problem 1. *Is it true that $Sp\big(Peb^d_G[\vee] \vdash 0\big) = \Omega\big(\textsf{BW-Peb}(G)\big)$ for any single-sink fan-in 2 DAG G provided that $d \geq 2$?*

Given that we know that this bound holds for a reasonably large class of graphs, we would find it surprising if it was not true in general.

Another question, for which we have much less intuition, is whether it holds that the refutation clause space of pebbling contradictions is in fact asymptotically lower-bounded not by black-white pebbling price but by *black* pebbling price.

As has been hinted at in, for instance, Section 9.1, it seems that a resolution refutation really cannot mimic a black-white pebbling in such a way that the number of clauses is upper-bounded by the number of pebbles. If one tries to transform a black-white pebbling into a resolution refutation along the lines of Section 9.1, it appears that the cost for white pebbles measured in terms of clauses very easily becomes exponential rather than linear. The litmus test for this question would be to study families of DAGs as in [47], where the black and black-white pebbling prices differ asymptotically, and determine the refutation clause space of, say, XOR-pebbling contradictions over such DAGs.

Open Problem 2. *Can it be that* $Sp\big(Peb^d_G[\oplus] \vdash 0\big) = \Omega\big(\mathsf{Peb}(G)\big)$ *for any single-sink fan-in 2 DAG* G *provided that* $d \geq 2$*? Or is there a family of DAGs* $\{G_n\}_{n=1}^{\infty}$ *such that* $Sp\big(Peb^d_G[\oplus] \vdash 0\big) = \mathrm{o}\big(\mathsf{Peb}(G)\big)$*? In particular, what is the refutation clause space of XOR-pebbling contradictions over the graphs in [47]?*

A third, slightly curious aspect of our results is that the bounds in Chapters 8 and 9 provide separations for k-CNF formulas only for $k \geq 4$, and in Chapter 10 we even have to choose $k \geq 6$ to find k-CNF formula families that optimally separate space and length. We know from [39] that any 2-CNF formula is refutable in constant clause space, but should there not be 3-CNF formulas for which we could prove similar space-length separations?

Given any CNF formula F, we can transform it to a 3-CNF formula by rewriting every clause $C = a_1 \vee \ldots \vee a_n$ in F wider than 3 as a conjunction of 3-clauses

$$\overline{y}_0 \wedge \bigwedge_{1 \leq i \leq n} (y_{i-1} \vee a_i \vee \overline{y}_i) \wedge y_n, \tag{12.1}$$

for some new auxiliary variables $y_0, y_1, \ldots, y_n$ unique for this clause C. Let us write $\widetilde{F}$ to denote the 3-CNF formula obtained from F in this way. It is easy to see that $\widetilde{F}$ is unsatisfiable if and only if F is unsatisfiable. Also, it is straightforward to verify that $L\big(\widetilde{F} \vdash 0\big) \leq L(F \vdash 0) + W(F) \cdot L(F)$ and $Sp\big(\widetilde{F} \vdash 0\big) \leq Sp(F \vdash 0) + \mathrm{O}\big(1\big)$.

It would seem like a fruitful idea to rewrite pebbling contradictions $Peb^d_G[f]$ for suitable functions f as 3-CNF formulas $\widetilde{Peb}^d_G[f]$ and study the space complexity of such formulas. For this to work, we would need *lower* bounds on the refutation clause space of $\widetilde{F}$ in terms of the refutation clause space of F, however.

Open Problem 3. *Is it true that* $Sp\big(\widetilde{Peb}^2_G[\oplus] \vdash 0\big) \geq \mathsf{BW\text{-}Peb}(G)$*? In general, can we prove lower bounds on* $Sp\big(\widetilde{F} \vdash 0\big)$ *in terms of* $Sp(F \vdash 0)$*, or are there counter-examples where the two measures differ asymptotically?*

We have not worked on the questions in Open Problem 3, but we would suspect that the answer to the first one is “yes.”

12.2 Space Lower Bounds for Stronger Proof Systems

It would be interesting to see whether our lower bounds can be extended to stronger proof systems than resolution. One very natural candidate would be the k-DNF resolution proof systems $\mathfrak{R}(k)$ introduced by Krajíček [50], where the lines in the proofs are k-DNF formulas instead of clauses and one can "resolve" over up to k variables simultaneously.

It is easy to prove the generalization of Theorem 5.8 that pebbling contradictions of degree d can be refuted in space $\mathit{Sp}_{\mathfrak{R}(k)}\bigl(\mathit{Peb}^d_G \vdash 0\bigr) = \mathrm{O}(1)$ in k-DNF resolution if $d \leq k$. For pebbling degree $d > k$, one could argue that it seems plausible that k-DNF resolution should be hard pressed to do anything better with Peb^d_G than ordinary resolution (i.e., 1-DNF resolution) can do with Peb^2_G. But although the difference between resolution and k-DNF resolution might appear small, going from disjunctive clauses to 2-DNF formulas, or more generally from k-DNFs to $(k{+}1)$-DNFs, increases the proof power exponentially [77]. And while many lower bounds have been proven on k-DNF resolution proof length, for instance, in [3, 8, 9, 71, 77], it seems that the tools developed in these papers cannot be used to obtain lower bounds on space.

A careful reading of our proofs reveals that the only place where we actually use that the configurations in the derivations contain disjunctive clauses is when we prove that the cost of the induced pebbles provide a lower bound for the number of clauses in the configuration inducing them. The proofs that resolution derivations induce pebblings (of one sort or another) work just as well for derivations that use any sound derivation rules and operate with configurations containing arbitrary logical formulas. The main difficulty if one tries to prove a lower bound on k-DNF resolution refutation space for standard OR-pebbling contradictions $\mathit{Peb}^d_G[\vee]$ appears to be that one needs an analogue of Theorem 8.27 for minimally unsatisfiable sets of k-DNF formulas, with a strong lower bound on the number of k-DNF formulas in terms of the number of variables. This result should then be plugged into Theorem 8.29 or Theorem 9.16 to yield a lower bound for k-DNF resolution refutation space. Unfortunately, to the best of our knowledge no such bounds for minimally unsatisfiable sets of k-DNF formulas have been shown, and we do not even have a clear intuition as to exactly what such an analogous result should look like.

Given our new results in Chapter 10, one could of course try with, for instance, XOR-pebbling contradictions $\mathit{Peb}^d_G[\oplus]$ instead. However, it seems that also in this case some additional ideas are needed to make the proof work.

We believe that both OR-pebbling contradictions $\mathit{Peb}^{k+1}_G[\vee]$ and XOR-pebbling contradictions $\mathit{Peb}^{k+1}_G[\oplus]$ should separate k-DNF resolution and $(k{+}1)$-DNF resolution with respect to space.

Open Problem 4. *If we fix k, does it hold for pebbling contradiction of degree $k{+}1$ that $\mathit{Sp}_{\mathfrak{R}(k+1)}\bigl(\mathit{Peb}^{k+1}_G[f] \vdash 0\bigr) = \mathrm{O}(1)$ but $\mathit{Sp}_{\mathfrak{R}(k)}\bigl(\mathit{Peb}^{k+1}_G[f] \vdash 0\bigr) = \Omega(\mathsf{BW\text{-}Peb}(G))$ for f chosen to be or $\vee$, exclusive or $\oplus$, or any other suitable function?*

If this bound, or any non-constant lower bound, on the k-DNF resolution space $Sp_{\mathfrak{R}(k)}(Peb_G^{k+1}[f] \vdash 0)$, could be proven, this would establish that the k-DNF resolution proof systems form a strict hierarchy with respect to space. Currently, all that is known is the separation result in [38] for the restricted case of tree-like k-DNF resolution. Also, understanding the question about the structure of minimally unsatisfiable sets of k-DNF formulas that arises when one studies OR-pebbling contradictions $Peb_G^d[\vee]$ might be an interesting (and challenging) combinatorial problem in its own right.

12.3 Trade-off Questions for Resolution

We also want to discuss some questions that are, perhaps, in a sense more related to Theorem 2.8 and the other trade-off results presented in Chapter 7.

Recall that we know from [21] (see Theorem 4.19) that short resolution refutations imply the existence of narrow refutations, and that in view of this an appealing proof search heuristic is to search exhaustively for refutations in minimal width. One serious drawback of this approach, however, is that there is no guarantee that the short and narrow refutations are the same one. On the contrary, the narrow refutation π' resulting from the proof in [21] is potentially exponentially longer than the short proof π that we start with. However, we have no examples of formulas where the refutation in minimum width is actually known to be substantially longer than the minimum-length refutation. Therefore, it would be valuable to know whether this increase in length is necessary. That is, is there a formula family which exhibits a length-width trade-off in the sense that there are short refutations and narrow refutations, but all narrow refutations have a length blow-up (polynomial or superpolynomial)? Or is the exponential blow-up in [21] just an artifact of the proof?

Open Problem 5. *If F is a k-CNF formula over n variables refutable in length L, is it true that there is always a refutation π of F in width $W(\pi) = \mathrm{O}\big(\sqrt{n \log L}\big)$ with length no more than, say, $L(\pi) = \mathrm{O}(L)$ or at most* $\mathrm{poly}(L)$*?*

A similar trade-off question can be posed for clause space. Given a refutation in small space, we can prove using [10] (see Theorem 4.22) that there must also exist a refutation in short length. But again, the short refutation resulting from the proof is not the same as that with which we started. For concreteness, let us fix the space to be constant. If a polynomial-size k-CNF formula has a refutation in constant clause space, we know that it must be refutable in polynomial length. But can we get a refutation in both short length and small space simultaneously?

Open Problem 6. *Suppose that $\{F_n\}_{n=1}^{\infty}$ is a family of polynomial-size k-CNF formulas with refutation clause space $Sp(F_n \vdash 0) = \mathrm{O}(1)$. Does this imply that there are refutations $\pi_n : F_n \vdash 0$ simultaneously in length $L(\pi_n) = \mathrm{poly}(n)$ and clause space $Sp(\pi_n) = \mathrm{O}(1)$?*

Or can it be that restricting the clause space, we sometimes have to end up with really long refutations? And what happens if the refutation clause space is small but slowly growing? We would like to know what holds in this case, and how it relates to the trade-off results for variable space in [45].

Finally, we note that all bounds on clause space proven so far is in the regime where the space $Sp(\pi)$ is less than the number of clauses $|F|$ in F. This is quite natural, since [39] proved that the size of the formula is an upper bound on the minimal clause space needed, as was mentioned in Section 4.3.

Such lower bounds on space might not seem too relevant to clause learning algorithms, since the size of the cache in practical applications usually will be very much larger than the size of the formula. For this reason, it seems to be a highly interesting problem to determine what can be said if we allow extra clause space. Assume that we have a CNF formula F of size roughly n refutable in length $L(F \vdash 0) = L$ for L suitably large (say, $L = \mathrm{poly}(n)$ or $L = n^{\log n}$ or so). Suppose that we allow clause space more than the minimum $n + \mathrm{O}(1)$, but less than the trivial upper bound $L/\log L$. Can we then find a resolution refutation using at most that much space and achieving at most a polynomial increase in length compared to the minimum?

Open Problem 7 ([17]). *Let F be any CNF formula with $|F| = n$ clauses (or $|Vars(F)| = n$ variables). Suppose that $L(F \vdash 0) = L$. Does this imply that there is a resolution refutation $\pi : F \vdash 0$ in clause space $Sp(\pi) = \mathrm{O}(n)$ and length $L(\pi) = \mathrm{poly}(L)$?*

If so, this could be interpreted as saying that a smart enough clause learning algorithm can potentially find any short resolution refutation in reasonable space (and for formulas that cannot be refuted in short length we cannot hope to find refutations efficiently anyway).

We conclude with a couple of comments on clause space versus clause learning.

Firstly, we note that it is unclear whether one should expect any fast progress on Open Problem 7, at least if if our experience from the case where $Sp(\pi) \leq |F|$ is anything to go by. Even in this "low-end regime," establishing lower bounds on space for formulas easy with respect to length has been quite challenging. However, it certainly cannot be excluded that problems in the range $Sp(\pi) > |F|$ might be approached with different and more successful techniques, or perhaps even by developing the ideas in Chapters 10 and 11 further.

Secondly, we would like to raise the question of whether, in spite of what was just said before Open Problem 7, lower bounds on clause space can nevertheless give indications as to which formulas might be hard for clause learning algorithms and why.

Suppose that we know for some CNF formula F that $Sp(F \vdash 0)$ is large. What this tells us is that any algorithm, even a non-deterministic one making optimal choices concerning which clauses to save or throw away at any given point in time, will have to keep a fairly large number of "active" clauses in memory in order to carry out the refutation. Since this is so, a real-life deterministic proof search algorithm, which has no sure-fire way of knowing which clauses are the right ones to concentrate on at any given moment, might have to keep working on a lot of extra clauses in order to be sure that the fairly large critical set of clauses needed to find a refutation will be among the "active" clauses.

Intriguingly enough, pebbling contradictions might in fact be an example of this. We know that these formulas are very easy with respect to length and width, having constant-width refutations that are essentially as short as the formulas themselves. But in [74], it was shown that state-of-the-art clause learning algorithms can have serious problems with even moderately large pebbling contradictions. Namely, the "grid pebbling formulas" in [74] are precisely our standard OR-pebbling contradictions of degree $d = 2$ over pyramids.

Although we are certainly not arguing that this is the whole story—it was also shown in [74] that the branching order is a critical factor, and that given some extra structural information the algorithm can achieve an exponential speed-up—we wonder whether the high lower bound on clause space can nevertheless be part of the explanation. It should be pointed out that pebbling contradictions are the only formulas we know of that are really easy with respect to length and width but hard for clause space. And if there is empirical data showing that for these very formulas clause learning algorithms can have great difficulties finding refutations, it might be worth investigating whether this is just a coincidence or a sign of some deeper connection.

Bibliography

[1] Douglas Adams. *The Restaurant at the End of the Universe*. Pan Macmillan, 1980.

[2] Ron Aharoni and Nathan Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *Journal of Combinatorial Theory*, 43:196–204, 1986.

[3] Michael Alekhnovich. Lower bounds for k-DNF resolution on random 3-CNFs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05)*, pages 251–256, May 2005.

[4] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002.

[5] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 448–456, May 2002.

[6] Michael Alekhnovich and Alexander A. Razborov. Resolution is not automatizable unless W[P] is tractable. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '01)*, pages 210–219, October 2001.

[7] Noga Alon and Michael Capalbo. Smaller explicit superconcentrators. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '03)*, pages 340–346, 2003.

[8] Albert Atserias and Maria Luisa Bonet. On the automatizability of resolution and related propositional proof systems. *Information and Computation*, 189 (2):182–201, March 2004.

[9] Albert Atserias, Maria Luisa Bonet, and Juan Luis Esteban. Lower bounds for the weak pigeonhole principle and random formulas beyond resolution. *Information and Computation*, 176(2):136–152, August 2002.

[10] Albert Atserias and Victor Dalmau. A combinatorical characterization of resolution width. In *Proceedings of the 18th IEEE Annual Conference on Computational Complexity (CCC '03)*, pages 239–247, July 2003. Journal version to appear in *Journal of Computer and System Sciences.*

[11] Sven Baumer, Juan Luis Esteban, and Jacobo Torán. Minimally unsatisfiable CNF formulas. *Bulletin of the European Association for Theoretical Computer Science*, 74:190–192, June 2001.

[12] Paul Beame. Proof complexity. In Steven Rudich and Avi Wigderson, editors, *Computational Complexity Theory*, volume 10 of *IAS/Park City Mathematics Series*, pages 199–246. American Mathematical Society, 2004.

[13] Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, 31 (4):1048–1075, 2002.

[14] Paul Beame, Henry Kautz, and Ashish Sabharwal. Understanding the power of clause learning. In *Proceedings of the 18th International Joint Conference in Artificial Intelligence (IJCAI '03)*, pages 94–99, 2003.

[15] Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present, and future. *Bulletin of the European Association for Theoretical Computer Science*, 65:66–89, June 1998.

[16] Eli Ben-Sasson. Size space tradeoffs for resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 457–464, May 2002.

[17] Eli Ben-Sasson. Personal communication, 2007.

[18] Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Structures and Algorithms*, 23(1):92–109, August 2003.

[19] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of treelike and general resolution. *Combinatorica*, 24(4):585–603, September 2004.

[20] Eli Ben-Sasson and Jakob Nordström. Short proofs may be spacious: An optimal separation of space and length in resolution. Submitted, April 2008.

[21] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001.

[22] Archie Blake. *Canonical Expressions in Boolean Algebra.* PhD thesis, University of Chicago, 1937.

[23] Maria Luisa Bonet, Carlos Domingo, Ricard Gavaldà, Alexis Maciel, and Toniann Pitassi. Non-automatizability of bounded-depth frege proofs. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity (CCC '99)*, pages 15–23, May 1999.

[24] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2000.

[25] Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, December 2001.

[26] Josh Buresh-Oppenheim and Toniann Pitassi. The complexity of resolution refinements. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 138–147, June 2003.

[27] Samuel R. Buss. Some remarks on the lengths of propositional proofs. *Archive for Mathematical Logic*, 34:377–394, 1995.

[28] Samuel R. Buss, editor. *Handbook of Proof Theory.* Elsevier Science, Amsterdam, 1998.

[29] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[30] Peter Clote and Evangelos Kranakis. *Boolean Functions and Computation Models.* Springer, 2002.

[31] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158, 1971.

[32] Stephen A. Cook. An observation on time-storage trade off. *Journal of Computer and System Sciences*, 9:308–316, 1974.

[33] Stephen A. Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, March 1979.

[34] Stephen A. Cook and Ravi Sethi. Storage requirements for deterministic polynomial time recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976.

[35] Dirk van Dalen. *Logic and Structure.* Universitext. Springer, Berlin, 3rd, augmented edition, 1994.

[36] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[37] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[38] Juan Luis Esteban, Nicola Galesi, and Jochen Messner. On the complexity of resolution with bounded conjunctions. *Theoretical Computer Science*, 321 (2-3):347–370, August 2004.

[39] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001.

[40] Juan Luis Esteban and Jacobo Torán. A combinatorial characterization of treelike resolution space. *Information Processing Letters*, 87(6):295–300, 2003.

[41] Zvi Galil. On resolution with clauses of bounded size. *SIAM Journal on Computing*, 6(3):444–459, 1977.

[42] John R. Gilbert and Robert Endre Tarjan. Variations of a pebble game on graphs. Technical Report STAN-CS-78-661, Stanford University, 1978. Available at `http://infolab.stanford.edu/TR/CS-TR-78-661.html`.

[43] James Gleick. A bug and a crash. *New York Times Magazine*, December 1996. Available at `http://www.around.com/`.

[44] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[45] Philipp Hertel and Toniann Pitassi. Exponential time/space speedups for resolution and the PSPACE-completeness of black-white pebbling. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*, pages 137–149, October 2007.

[46] John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *Journal of the ACM*, 24(2):332–337, April 1977.

[47] Balasubramanian Kalyanasundaram and George Schnitger. On the power of white pebbles. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 258–266, 1988.

[48] Henry Kautz and Bart Selman. The state of SAT. *Discrete Applied Mathematics*, 155(12):1514–1524, June 2007.

[49] Maria M. Klawe. A tight bound for black and white pebbles on the pyramid. *Journal of the ACM*, 32(1):218–228, January 1985.

[50] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1-3):123–140, 2001.

[51] Jan Krajíček, Pavel Pudlák, and Alan R. Woods. An exponential lower bound to the size of bounded depth frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7(1):15–40, 1995.

[52] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *Journal of Symbolic Logic*, 54(3):1063–1079, 1989.

[53] Oliver Kullmann. An application of matroid theory to the SAT problem. In *Proceedings of the 15th Annual IEEE Conference on Computational Complexity (CCC '00)*, pages 116–124, July 2000.

[54] Thomas Lengauer and Robert Endre Tarjan. Upper and lower bounds on time-space tradeoffs. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC '79)*, pages 262–277, May 1979.

[55] Thomas Lengauer and Robert Endre Tarjan. The space complexity of pebble games on trees. *Information Processing Letters*, 10(4/5):184–188, July 1980.

[56] Friedhelm Meyer auf der Heide. A comparison of two variations of a pebble game on graphs. *Theoretical Computer Science*, 13(3):315–322, 1981.

[57] The Millennium Problems of the Clay Mathematics Institute, May 2000. See `http://www.claymath.org/millennium/`.

[58] Cleve Moler. A tale of two numbers, 1995. Available at `http://www.mathworks.com/company/newsletters/news_notes/clevescorner/`.

[59] Jakob Nordström. Stålmarck's method versus resolution: a comparative theoretical study. Master's thesis, Stockholm University, 2001. Available at `http://www.csc.kth.se/~jakobn/research/`.

[60] Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution (Extended abstract). In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06)*, pages 507–516, May 2006.

[61] Jakob Nordström. A simplified way of proving trade-off results for resolution. Technical Report TR07-114, Electronic Colloquium on Computational Complexity (ECCC), September 2007.

[62] Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution. *SIAM Journal on Computing*, 2008. To appear. Preliminary version available at `http://www.csc.kth.se/~jakobn/research/`.

[63] Jakob Nordström and Johan Håstad. Towards an optimal separation of space and length in resolution (Extended abstract). In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, May 2008. To appear.

[64] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[65] Christos H. Papadimitriou and David Wolfe. The complexity of facets resolved. *Journal of Computer and System Sciences*, 37(1):2–13, 1988.

[66] Joachim Parrow. Programmeraren som schaman. *Forskning & Framsteg*, 2:14–19, February 1998. In Swedish. Available at `http://fof.se/?id=98214`.

[67] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[68] Nicholas Pippenger. Pebbling. Technical Report RC8258, IBM Watson Research Center, 1980. Appeared in Proceedings of the 5th IBM Symposium on Mathematical Foundations of Computer Science, Japan.

[69] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3:97–140, 1993.

[70] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, March 1999.

[71] Alexander A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. Manuscript. Available at the webpage `http://www.mi.ras.ru/~razborov/`, July 2003.

[72] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[73] Ashish Sabharwal. *Algorithmic Applications of Propositional Proof Complexity.* PhD thesis, University of Washington, Seattle, 2005.

[74] Ashish Sabharwal, Paul Beame, and Henry Kautz. Using problem structure for efficient clause learning. In *6th International Conference on Theory and Applications of Satisfiability Testing (SAT '03), Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2004.

[75] The international SAT Competitions. `http://www.satcompetition.org`.

[76] Nathan Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):482–537, December 2007.

[77] Nathan Segerlind, Samuel R. Buss, and Russell Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004.

[78] Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33(3):277–280, May 1996.

[79] Jacobo Torán. Lower bounds for space in resolution. In *Proceedings of the 13th International Workshop on Computer Science Logic (CSL '99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 362–373. Springer, 1999.

[80] Jacobo Torán. Space and width in propositional resolution. *Bulletin of the European Association for Theoretical Computer Science*, 83:86–104, June 2004.

[81] Grigori Tseitin. On the complexity of derivation in propositional calculus. In A. O. Silenko, editor, *Structures in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125. Consultants Bureau, New York-London, 1968.

[82] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1): 209–219, January 1987.

[83] Alasdair Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1(4):425–467, 1995.