Stålmarck's Method versus Resolution: A Comparative Theoretical Study

Jakob Nordström

September 7, 2001

Abstract

This Master's thesis presents a comparative analysis of Stålmarck's proof method and resolution from a theoretical perspective.

We give (to our knowledge) the first complete explicit formal description of the dilemma proof system underlying Stålmarck's method. Based on this description we prove a number of simulation and separation results between different subsystems of dilemma (defined by restrictions on possible branching assumptions and rules for merging the results derived in distinct branches).

The key result of the thesis is that dilemma depth translates into resolution width. More precisely, a dilemma refutation in depth d and length L of a k-CNF formula F can be transformed to a resolution refutation of F in width $\mathrm{O}(kd)$ and length $\left(Lk^d\right)^{\mathrm{O}(1)}$.

From this depth-width relation it follows that for k-CNF formulas with k fixed, resolution p-simulates dilemma restricted to minimum-depth proofs. Furthermore, the running time of the minimum-width proof search algorithm suggested by Ben-Sasson and Wigderson is shown to be polynomial in the running time of Stålmarck's method.

Finally, using results by Beame et al. we show that if F is a uniformly random 3-CNF formula with Δn clauses on n variables, then with high probability it holds for the dilemma hardness $H_{\mathcal{D}}(F)$ that $\Omega\left(n/\Delta^{2+\epsilon}\right) \leq H_{\mathcal{D}}(F) \leq \mathrm{O}(n/\Delta)$ (where ϵ is an arbitrarily small but positive constant).

Referat

Stålmarcks metod versus resolution: En komparativ teoretisk studie.

Denna examensarbetesrapport presenterar en komparativ analys av Stålmarcks bevismetod och resolution ur ett teoretiskt perspektiv.

Vi ger den (såvitt vi vet) första fullständiga explicita formella beskrivningen av bevissystemet dilemma, på vilket Stålmarcks metod är baserad. Utgående ifrån denna beskrivning bevisar vi ett antal simulerings- och separationsresultat mellan olika delsystem av dilemma (definierade genom inskränkningar av reglerna för antaganden vid förgreningar och för att sammanfoga resultat härledda i olika grenar).

Huvudresultatet i denna rapport är att djup i dilemma svarar mot bredd i resolution. Mer precist formulerat kan en dilemmarefutation i djup d och längd L av en k-CNF-formel F överföras till en resolutionsrefutation av F i bredd $\mathrm{O}(kd)$ och längd $\left(Lk^d\right)^{\mathrm{O}(1)}$.

Från denna relation mellan djup och bredd följer det att för k-CNF-formler med k fixerat p-simulerar resolution dilemma begränsat till bevis i minimalt djup. Vidare visas att exekveringstiden för den algoritm för sökning av bevis i minimal bredd som föreslagits av Ben-Sasson och Wigderson är polynomiell i exekveringstiden för Stålmarcks metod.

Slutligen visar vi med användande av resultat av Beame et al. att om F är en likformigt fördelad slumpmässig 3-CNF-formel med Δn klausuler över n variabler så gäller med hög sannolikhet för svårighetsgraden i dilemma $H_{\mathcal{D}}(F)$ att $\Omega\left(n/\Delta^{2+\epsilon}\right) \leq H_{\mathcal{D}}(F) \leq \mathrm{O}(n/\Delta)$ (där ϵ är en godtyckligt liten men positiv konstant).

Preface

This is a Master's thesis in Computer Science on the First Degree Programme in Mathematics and Computer Science at Stockholm University. The thesis reports research done at Prover Technology AB under the supervision of professor Gunnar Stålmarck. The supervisor and examiner at the Department of Numerical Analysis and Computing Science, Royal Institute of Technology was professor Johan Håstad.

I would like to thank Johan Håstad for providing valuable insights and pointing out the connections that lie at the foundation of this report. Also, I want to thank Gunnar Stålmarck for giving me the opportunity of doing my Master's project at Prover Technology, for all the interesting and inspiring discussions and for providing the subject matter of this thesis in the first place. Finally, special thanks go to Paul Beame and Toniann Pitassi for kindly taking the time to answer my questions about their articles on bounds on length of refutations of random k-CNF formulas.

Contents

Chapter 1

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 designed for the predecessor Ariane 4 but useless for Ariane 5. To quote the inquiry board report, this piece of program 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. Because of this, the program received unexpectedly large horizontal velocity data, which caused an overflow error. Since there was a large safety margin for these values in Ariane 4, no error checking had been included to take care of a possible overflow, so the program crashed. The ensuing error message was interpreted by the onboard computer as flight data, and as a result the rocket made "an abrupt course correction that was not needed, compensating for a wrong turn that had not taken place", broke up and exploded (the quote is from [29], which contains a very readable analysis of the launch failure and its wider implications). 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 example of many 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 one does not have to go further than the most common word processors and spreadsheets in use today.

The contemporary hardware development industry experiences analogous problems. Perhaps the most well-known illustration of this is the embarrassing flaw in Intel's Pentium microprocessor in October 1994. Then it was discovered that because of some missing entries in a look-up table in the floating-point unit, division operations could in rare cases produce wrong results. 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 (see for instance [42] for a chronological account of the story). The Pentium division bug became a wake-up call that the existing process for chip development was inadequate.

Intel is not alone 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.

Today, verification takes anywhere from 40% to 70% of the total time spent designing a new chip [45]. On some projects in the hardware and telecommunication industries, the cost of simulation and testing is nearing 50% of overall project funding [9]. But simulation and testing can never provide full coverage of all possible cases in increasingly complex designs.

One proposed solution to the problems discussed above is the adoption of *formal methods* in software and hardware design. If the methods for specifying and designing systems are formalized, it becomes possible to use mathematical tools to prove theoretically 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 decade.

In this Master's thesis, we address one of the fundamental questions in formal methods, namely *automated theorem proving* and its theoretical foundations in the subject of *proof theory*. This first chapter is an attempt to give an short, informal introduction to all three above-mentioned subjects.

1.1 Shortly about Formal Methods

First of all, we have to make clear what we mean by a "formal method". The Free On-line Dictionary of Computing (`http://foldoc.doc.ic.ac.uk/`) describes formal methods as:

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

A more detailed explanation of the term is given in the UK Ministry of Defence standard for safety-critical software [46], which defines a formal method as:

> *a software specification and production method, based on mathematics, that comprises: a collection of mathematical notations addressing the specification, design and development processes of software production; a well-founded logical inference system in which formal verification proofs and proofs of other properties can be formulated; and a methodological framework within which software may be developed from the specification in a formally verifiable manner.*

This second definition of formal methods is a rather ambitious one. In practice, not all of the above mentioned components need be present.

Before going into more detail, it is important to note that formal methods are concerned with *models* of systems and that the methods can only be applied to these models, not to the real systems themselves. Thus, formal methods per se is no guarantee for correctness. Nevertheless, the use of formal methods can greatly increase understanding of a system by revealing inconsistencies and ambiguities that might otherwise remain undetected.

There are two main approaches to formal methods: specification oriented and verification oriented.

Specification is the process of describing a system and its desired properties. A *specification oriented formal method* provides a language with mathematically defined syntax and semantics for this process. The kind of properties specified can be for example functional behaviour, timing behaviour, performance characteristics or internal structure.

The main benefit of specification is that it forces the developer to think through the system carefully, and in this way helps him or her gain a deeper understanding of the system being specified. Through this process the developer can uncover design flaws. What is more, since the specification is written in a language with formally defined meaning, it can itself be checked for consistency or used to derive properties about the system. The purpose of such analysis is to *validate* the specification, that is to make sure that it matches the intentions of the designer.

Verification goes one step beyond specification. The goal of a *verification oriented formal method* is to make sure that an implementation conforms to some specification. Two well-established approaches to verification are model checking and theorem proving.

Model checking is a technique that models the system as a finite state machine and checks that desired properties hold by exhaustively searching through the state space of the model. This search is guaranteed to terminate since the search space is finite, but the challenge is to devise algorithms and data structures that allow handling of large state spaces. Model checking has been used primarily in hardware and protocol verification, but is also being applied to analysis of software system specifications.

The desired properties can be specified in temporal logic, after which an efficient search algorithm is used to examine whether the given finite state machine is a model for the specification or not. Another variant is to give the specification as an automaton and compare it to the system, also modelled as an automaton, to find out if the behaviour of the latter conforms to that of the former.

Some advantages of model checking is that it is completely automatic, that it can be used to check partial specifications and so provide useful information about correctness even if a system has not been completely specified, and that it produces counterexamples and thus can be used to debug subtle errors in the design. The main drawback of the technique is that for real world problems, the search space can become very big. This is usually referred to as the *state explosion problem*. One way of coping with this problem is to avoid representing the state space explicitly by encoding large sets of states symbolically in a compact format, so called *symbolic model checking*.

Theorem proving is a technique where both the system and the specification are expressed as formulas in some mathematical logic. The logic in question is given by a formal system, which defines a set of axioms and a set of inference rules. The desired relationship between the implementation and the specification (for instance that the implementation logically implies the specification, or that implementation and specification are logically equivalent) is then considered as a theorem in this logic. Theorem proving is the process of finding proofs of such theorems (with the assistance of a computer theorem prover). This technique is increasingly being used today in the verification of safety-critical properties of hardware and software systems.

Theorem provers can be classified in a wide spectrum ranging from highly automated general-purpose proof engines to special-purpose interactive systems. In contrast to model checkers, theorem provers can deal directly with infinite state spaces, for example by using induction techniques.

Different formal methods can be combined to describe and analyze different aspects of a complex system, since no single method is likely to be suitable for analysis of every aspect of a system. Also, formal methods can be integrated into the overall system development process to complement methods of a less formal nature. We conclude our short exposition of formal methods by giving three examples of this:

- Formal methods can be used in *requirement analysis* to convert the imprecise ideas of a customer into precise system requirements.

- Formal methods can be used for *refinement*, the reverse process of verification. Refinement is to take one level of specification (or implementation) and use it to synthesize a lower-level specification (or implementation).

- Formal methods can be used in *testing*. Testing is one of the costliest parts in all software projects. It is possible to make the testing procedures more efficient by using formal methods for instance to generate test suits based on the information given by a formal specification.

Needless to say, it is impossible to give an exhaustive treatment of formal methods on a couple of pages, especially considering that different authors interpret the term differently. Below we give some suggestions for further reading.

A nice informal introduction to formal methods (in Swedish) is [40]. This section relies heavily on [17], which is an informative survey of the subject with theory and practical examples. [9] is a more practically applied survey of formal methods in a broader sense which concentrates on the reasons for their industrial success or failure. [48] is an introduction to the use of formal methods in hardware verification (for the ambitious reader, [31] is a more extensive survey of this field). Finally, we are indebted to the introductory material in [8] (which also contains a short but interesting historic overview).

1.2 Shortly about Automated Theorem Provers

Automated theorem provers, also called *proof engines* or *automated reasoning systems*, are computer programs which perform automated logical deduction within the framework of some logic.

Often, a natural choice of logic in this context is first-order predicate logic. An interesting special case of predicate logic is propositional logic, which deals with the relationship between propositions, but, unlike predicate logic, not their internal structure. Not seldom problems in predicate logic can be reduced to propositional logic in one way or another. In this thesis, we will restrict our attention to automated theorem provers in propositional logic.

A formula F in propositional logic is a theorem if and only if all truth value assignments to the variables of F make the formula true, i.e. if there is no way to satisfy the negation of F. In the opposite direction, if we find a satisfying assignment of the negation of a formula F, the same assignment falsifies F and thus proves that this formula cannot be a theorem. Thus theorem proving is very closely related to the problem of determining propositional satisfiability, usually abbreviated as SAT. Depending on the perspective, one can reason either in terms of automated theorem provers or in terms of propositional satisfiability procedures, also referred to as SAT-based methods. We will not make any distinction between theorem provers and SAT-based methods in the following, but consider them to be two sides of the same coin.

Looking back at section 1.1, the connection between automated theorem provers and the theorem proving approach to formal verification is perhaps rather obvious. But theorem provers are interesting also in the context of model checking. As was mentioned above, for realistic design models the number of states of the system can be very large, which makes explicit traversal of the state space infeasible. One way of tackling this difficulty is to use a symbolic representation of the state space and reduce the resulting symbolic model checking problem to a propositional satisfiability problem. The latter problem can then be solved by an automated theorem prover.

Formal verification is one application area for the automation of proof, but the list of applications is much larger. Apart from hardware and software design analysis and verification, automated theorem provers are used among other things for scheduling problems, in artificial intelligence and even to prove results in theoretical mathematics.

There are a number of different approaches to automated theorem proving. The first theorem provers originated in the 1950s in the context of artificial intelligence research. The goal was to invent a “thinking machine”, to simulate the human process of deduction. Parallel development in research concentrated on methods rooted more strongly in the traditions of mathematical logic led to Robinson's resolution principle [47]. Since then, most theorem provers have been based either on some variant of resolution or on semantic tableaux [50]. During the last ten years or so, provers based on binary decision diagrams, or BDDs [12, 13] have also gained in popularity.

Another promising approach in automated theorem proving is a relatively new proof search algorithm invented by Gunnar Stålmarck in the mid 1980s [49]. Stålmarck's method, as it is called, was patented in 1989 and is the basis of the proof engines offered by Prover Technology AB, founded in Stockholm, Sweden the same year. The distinctive feature of Stålmarck's method is that in contrast to for instance BDD packages, it can cope with very large formulas provided that they are *easy* according to a formally defined measure which will be discussed later in this thesis. Empirically, important cases of real world problems give rise to large but easy formulas.

Prover Technology's main markets are Electronic Design Automation (EDA) and Computer Aided Software Engineering (CASE), and its products have also been successfully applied to system development in avionics, nuclear power generation, railroads, automotive industry and telecommunications. Furthermore, Stålmarck's method has been the subject of a number of industrial and academic research projects. We refer to [49] or `www.prover.com` for an updated list of references.

1.3 Shortly about Proof Theory

The main focus of this Master's thesis is a comparison of Stålmarck's theorem proving algorithm with algorithms based on resolution. In particular, we prove bounds on running time by deriving theoretical bounds on the *proofs* which can be found by these algorithms.

All theorem provers, regardless of whether they actually produce a written proof or not, explicitly or implicitly define a system in which proofs are searched for and format rules which determine what proofs in this system look like. Such "systems" with "format rules" are called *proof systems* and are studied in the subject of *proof theory*.

Slightly more formally, a proof system can be described as a format for presenting proofs together with an efficient algorithm for checking that proofs presented in this format are valid. An automated theorem prover can then be seen as an algorithm for searching for proofs in the corresponding proof system.

The design of a proof system can have a dramatic impact on the performance of proof search algorithms in the system. If a proof system has the property that certain formulas have no small proofs satisfying the format rules, then no proof search algorithm based on this system can be efficient for these particular formulas (since the running time of the algorithm must be at least as large as the proof it eventually finds). Putting it differently, lower bounds on proofs in proof systems give theoretically proven lower bounds on the running time of corresponding automated theorem provers. Also, theoretical upper bounds on proof size in a system can give upper bounds on the running time of a proof search algorithm, provided that the algorithm can be shown to search for proofs in the system in an efficient manner.

In view of the above, the goal of automated theorem proving can be expressed as constructing powerful proof systems and efficient algorithms to search for proofs in these systems.

The most powerful proof systems we can reasonably expect are systems where the sizes of proofs are at most as large as some polynomial expression in the size of the formula proven (where the polynomial expression does not depend on the particular formula in question but only on the proof system). Such proof systems are called *polynomially bounded*. It is not known whether there exist any polynomially bounded systems at all for propositional logic. For example, the proof systems studied in this thesis are known *not* to be polynomially bounded (more about this below).

Regardless of whether there exist polynomially bounded proof systems or not, we are not very much helped by a strong proof system if there are no efficient algorithms for searching for proofs in the system. Proof systems which have efficient proof search algorithms are called *automatizable*. The efficiency of a proof search algorithm is measured by how fast it can find a proof of a formula in terms of the smallest possible proof for this formula in the system. This means that the fact that a proof system is automatizable does not necessarily imply that it is powerful, only that there are algorithms which so to speak realize the full potential of the system, however large or small this potential may be.

Sadly enough, the results of contemporary research indicate that we most probably cannot expect a proof system to be both powerful and automatizable. But this does not exclude the possibility that we can find proof systems and algorithms with the help of which most problems which turn up *in practice* can be solved efficiently. Therefore, even a non-automatizable proof system which is not polynomially bounded can be of great interest if it has small proofs which can be found efficiently for most formulas resulting from real world problems.

Proof theory is also a subject of independent interest, quite apart from its connection to automated theorem proving. This is mainly because of the fact that it is intimately related to fundamental questions in complexity theory.

Loosely put, complexity theory is the discipline of computer science which measures the resources needed to solve problems on a computer, for example the amount of time or memory that a computation demands. Informally, from the point of view of complexity theory problems are considered easy, or tractable, if they can be solved by computer algorithms that run in polynomial time; that is, for a problem of size n, the time or number of steps needed to find the solution is bounded by a polynomial function of n. Problems which can only be solved by algorithms that require an amount of time exponential in the problem size n are considered hard, or intractable. Polynomial-time algorithms are regarded as efficient, while exponential-time algorithms are considered inefficient, because the execution times of the latter grow much more rapidly as the problem size increases.

We say that problems which have polynomial-time solutions are members of the *complexity class* P. A very much simplified way of describing P is to say that it is the class of problems which can be solved on a computer not only in theory but also in practice.[1]

Another important class of problems is the class known as NP. Problems in NP can be described by the property that they can be hard to solve (at least empirically), but if we are somehow given a proposed solution it is easy to verify whether it is correct or not. One natural example of such a problem is integer factorization. Given a large integer it can be hard to compute its factors, but it is easy to multiply together the factors in a suggested factorization to check that it is correct. More precisely, a problem is said to be in the complexity class NP, or to be solvable in nondeterministic polynomial time, if a guessed solution can be verified in polynomial time; nondeterministic means that unfortunately, there are not necessarily any good rules for making the needed lucky guess.

Note that not all problems in NP are hard. In particular, all problems in P are also members of NP, since for such problems we can always "guess and verify" a solution in polynomial time by actually computing it. A less trivial example of a member of NP is the integer factorization problem discussed above. The hardest problems in NP are the so called NP-*complete* ones. Two examples of NP-complete problems are the travelling salesman problem *TSP* (given a set of towns and the distances between them, determine the shortest path starting from a given town, passing through all the other towns and returning to the first town) and the satisfiability problem *SAT* (given a propositional logic formula F, decide if there is some assignment to the variables in F which makes the formula true).

[1]This is of course an irresponsibly oversimplified and arguably erroneous way of describing P, but we still find it appropriate in this very informal setting.

Many important real-life problems have been shown to be NP-complete, but so far no efficient algorithms for solving such problems have been invented. It is not known whether any efficient algorithms exist at all. However, all NP-complete problems are mutually related in the sense that they can be reformulated in terms of each other. Therefore, if there is some effective algorithm for one of the NP-complete problems (i.e. an algorithm which runs in time polynomial in the input), then this algorithm can be used to solve *all* problems in NP efficiently.

Determining whether NP-complete problems are tractable or intractable, that is whether $\mathsf{P} = \mathsf{NP}$ or not, remains one of the most important questions in theoretical computer science. This is also a question of great practical importance, since some widely used modern cryptosystems are based on the assumption that factorization while probably not NP-complete still is not a tractable problem. Most computer scientist suspect that $\mathsf{P} \neq \mathsf{NP}$. This has not been proven, though, and there are proponents of the opposite view.[2]

The connection to proof theory is that one way of proving the inequality $\mathsf{P} \neq \mathsf{NP}$ would be to show that there cannot exist any polynomially bounded proof systems for propositional logic. This result is stronger than $\mathsf{P} \neq \mathsf{NP}$, however, so the existence of polynomially bounded proof systems would not imply $\mathsf{P} = \mathsf{NP}$.

1.4 Organization of This Thesis

The organization of the rest of this thesis is as follows. In chapter 2, we give a formal presentation of the subject of proof theory, including the resolution proof system. We continue our study of resolution in chapter 3, where we present some recent results. A detailed exposition of Stålmarck's method, and the dilemma proof system in terms of which is is defined, follows in chapter 4. In chapter 5, we develop tools for proving results about the dilemma proof system and Stålmarck's method and their relation to resolution and resolution-based proof methods. Chapter 6 contains the results proved in this thesis. These results are summarized in chapter 7, where we also note some open questions and give suggestions for further research. Appendix A contains some missing details regarding the dilemma proof system left out in chapters 4 and 5. In appendix B, finally, we provide detailed proofs of two of the theorems in chapter 6.

[2] In fact, Stålmarck's method was invented as a result of a failed proof of $\mathsf{P} = \mathsf{NP}$.

Chapter 2

Proof Theory

In section 1.3 we gave a short and popular presentation of proof theory in the context of formal methods and automated theorem proving. This chapter is intended as a more formal (if selective) introduction to proof theory in general and resolution in particular. Our presentation is based on material from [5, 14, 57], to which the reader is referred for more details.

2.1 Elementary Definitions

Although the subject matter of this chapter is logic, we start off by making clear our usage of asymptotic notation and terminology.

Definition 2.1 (Asymptotic notation) *We say that $f(n)$ is $\mathrm{O}(g(n))$ and write $f(n) = \mathrm{O}(g(n))$ if there exist $c \in \mathbb{R}^+$ and $n_0 \in \mathbb{N}$ such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.*

$f(n) = \Omega(g(n))$ if $\exists\, c \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

$f(n) = \Theta(g(n))$ if $f(n) = \mathrm{O}(g(n))$ and $f(n) = \Omega(g(n))$.

$f(n) = \mathrm{o}(g(n))$ if $\forall\, c \in \mathbb{R}^+\ \exists\, n_0 \in \mathbb{N}$ such that $0 \leq f(n) < c \cdot g(n)$ for $n \geq n_0$.

$f(n) = \omega(g(n))$ if $\forall\, c \in \mathbb{R}^+\ \exists\, n_0 \in \mathbb{N}$ such that $0 \leq c \cdot g(n) < f(n)$ for $n \geq n_0$.

Definition 2.2 (Asymptotic terminology) *A function $f(n)$ is* polynomial *in n if $f(n) = \mathrm{O}\left(n^k\right)$ for some $k \in \mathbb{N}$. We will (strictly speaking incorrectly) usually write this as $f(n) = n^{\mathrm{O}(1)}$, using $\mathrm{O}(1)$ as a shorthand for an arbitrary non-negative constant or function limited by a constant.*

$f(n)$ is pseudo- *or* quasi-polynomial *in n if $f(n) = \mathrm{O}\big(\exp\big((\log n)^k\big)\big)$ for some $k \in \mathbb{N}$, which we will usually simplify as $f(n) = \exp\big((\log n)^{\mathrm{O}(1)}\big)$.*

$f(n)$ is subexponential *in n if $f(n) = \mathrm{o}\left(\exp(n^c)\right)$ for all $c \in \mathbb{R}^+$.*

$f(n)$ is superlogarithmic *in n if $f(n) = \omega(\log n)$.*

$f(n)$ is superpolynomial *in n if $f(n) = \omega\left(n^k\right)$ for all $k \in \mathbb{N}$.*

$f(n)$ is exponential *in n if $f(n) = \Omega\left(\exp(n^c)\right)$ for some $c \in \mathbb{R}^+$.*

Remark 2.3 We include definitions 2.1 and 2.2 since there seem to be variations in the literature concerning asymptotic notation and what is meant by terms such as "exponential", "subexponential", "quasi-polynomial" et cetera. The reason for our definitions of "exponential" and "subexponential" is that we want for example $\exp(\sqrt{n})$ to be exponential and definitely *not* subexponential (otherwise changing the parameter n of a problem polynomially could turn exponential problems into subexponential ones and vice versa).

In the rest of this section, we give some elementary definitions from propositional logic. We refer the reader to, for example, [22] for a fuller and more stringent treatment of (most of) the definitions, notation and terminology below.

We assume the existence of an infinite set *Vars* of boolean (or propositional logic) variables. We let the boolean variables range over $\{\bot, \top\}$ or $\{0, 1\}$, where we identify $\bot$ and 0 with *FALSE* and $\top$ and 1 with *TRUE* respectively. We usually let the letters x, y, z and w (possibly with indices) denote variables.

We use the traditional set of logical connectives: negation $\neg$, conjunction $\wedge$, disjunction $\vee$, implication $\rightarrow$ and bi-implication $\leftrightarrow$ (to avoid confusion, we will refer to $\leftrightarrow$ as "bi-implication" and reserve the term "equivalence" for the formula equivalence relation defined in chapter 4).

Definition 2.4 *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* $\big(F \wedge G\big), \big(F \vee G\big), \big(F \rightarrow G\big), \big(F \leftrightarrow G\big) \in X$,
- *if* $F \in X$ *then* $\big(\neg F\big) \in X$.

For convenience, we will usually omit the outermost pair of parentheses in a propositional logic formula. Also, we define $\neg$ to have higher precedence than $\wedge$ and $\vee$, which in turn have higher precedence than $\rightarrow$ and $\leftrightarrow$, and skip unnecessary parentheses whenever this order of precedence makes the intended meaning clear.

The set of variables of a formula F is denoted $\mathit{Vars}(F)$. The symbol $\doteq$ used in the definitions below denotes syntactic equality.

Definition 2.5 (Subformula) *For F and G arbitrary formulas in propositional logic, G is a* subformula *of F if*

- $G \doteq F$, *or*
- $F \doteq P \circ Q$ *(where* $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$*), and G is a subformula of P or Q, or*
- $F \doteq \neg P$ *and G is a subformula of P.*

The set of (true syntactic) subformulas of a formula F is denoted $\mathit{Sub}_{\doteq}(F)$.

More often, we will be interested in subformulas of F *and their negations.*

Definition 2.6 (Formula complement) *Let P be a formula in propositional logic. The* formula complement *(or just* complement*) P^C of P is defined by*

$$P^C = \begin{cases} Q & \text{if } P \doteq \neg Q \text{ for some } Q, \\ \neg P & \text{otherwise.} \end{cases}$$

Definition 2.7 *We define*

$$Sub\,(F) := Sub_{\doteq}\,(F) \cup \{G^C \mid G \in Sub_{\doteq}\,(F)\} \cup \{\bot, \top\}$$

to be the set of subformulas of a formula F as well as the complements of these formulas and the constants $\top$ and $\bot$.

In this thesis, we will abuse terminology slightly and refer to the set $Sub\,(F)$ as the "subformulas of F".

Definition 2.8 (Compound subformula) *Given a formula F, the set of* compound subformulas *of F, which we denote $Compound\,(F)$, is defined as*

$$Compound\,(F) := Sub\,(F) \setminus \big(\{\top, \bot\} \cup \{x, \neg x, \neg(\neg x), \ldots \mid x \in Vars(F)\}\big)$$

and we call G a compound subformula *of F if $G \in Compound\,(F)$.*

That is, compound subformulas are all subformulas other than truth value constants and unnegated or negated (possibly repeatedly) variables.

Given a formula F, a *valuation* on F is a function $\alpha : Sub_{\doteq}\,(F) \mapsto \{\bot, \top\}$ which respects the traditional semantics of $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. We say that F is

- *satisfiable* if there is an valuation α on F with $\alpha(F) = \top$,
- *valid* or *tautological* if all valuations on F satisfy F,
- *falsifiable* if there is an valuation α on F with $\alpha(F) = \bot$,
- *unsatisfiable* or *contradictory* if all valuations on F falsify F.

If a valuation 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 $TAUT$.

Every assignment of truth values to the variables in $Vars(F)$ defines a valuation α on F. In this way, any formula F induces a *boolean function* $f_F : Vars(F) \mapsto \{\bot, \top\}$ with $f_F\big(\alpha\big) = \alpha(F)$. At times, we will somewhat incorrectly let F denote the function f_F induced by F and speak of the "(boolean) function" F.

Definition 2.9 *For F a formula and $\mathcal{G} = \{G_1, \ldots, G_n\}$ a set of formulas, we say that $\mathcal{G}$* implies *F, denoted $\mathcal{G} \models F$, if every valuation satisfying all formulas $G \in \mathcal{G}$ satisfies F as well.*

In view of the paragraph just before the definition, we will use the same terminology and notation when F and $G_1, \ldots, G_n$ are boolean functions.

2.2 Propositional Proof Systems

Given a tautological formula F in propositional logic, what is the minimum size of a proof of F in a proof system $\mathcal{P}$? And what is the relation between minimal proofs proof π_i of F in different proof systems $\mathcal{P}_i$?

Before we can answer such questions, we have to decide what exactly constitutes a "proof" and what a "proof system" is. For example, since there are only finitely many truth value assignments to check, why not allow the formula F itself as a proof of the fact that F is a tautology? The key observation here is that a proof, unlike the tautology itself, should be easy to check. Also, in order to make this check we need to know the format in which the proof will be presented.

These considerations lead us to the following definition.

Definition 2.10 (Propositional proof system) *A* propositional proof system *is a polynomial-time computable binary predicate $\mathcal{P}$ satisfying the following property: for all propositional logic formulas F it holds that $F \in TAUT$ if and only if there exists a* proof π *of F such that $\mathcal{P}(F, \pi)$ is true.*

That is, we identify a proof system $\mathcal{P}$ with a procedure for checking the correctness of proofs π of tautologies F which has running time polynomial in F and π. (Anticipating the notation and terminology presented in section 2.3, we assume that F and π are strings in some alphabet Σ such that $PROP \subseteq \Sigma^*$).

The *complexity* of a propositional proof system is defined as the lowest bound on the size of proofs of tautological formulas as a function of the size of the formulas themselves.

Definition 2.11 (Size) *The* size $S(F)$ *of a formula F is defined to be the total number of symbols in F. The size $S(\pi)$ of a proof π in some propositional proof system $\mathcal{P}$ is the total number of symbols in π.*

Definition 2.12 (Complexity) *The* complexity *of a propositional proof system $\mathcal{P}$, which we denote $comp_{\mathcal{P}}$, is the smallest bounding function $g : \mathbb{N} \mapsto \mathbb{N}$ for which*

$$F \in TAUT \Leftrightarrow \exists\pi\ S(\pi) \leq g\big(S(F)\big) \wedge \mathcal{P}\big(F, \pi\big).$$

If a proof system is of polynomial complexity, it is said to be polynomially bounded *or* p-bounded*.*

It is not known whether there exist any p-bounded propositional proof systems. The answer to this question has far-reaching implications for computational complexity theory, as we will see in theorem 2.15 below.

The main tool for comparing proof systems is *p-simulation*.

Definition 2.13 (p-simulation) *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be propositional proof systems. $\mathcal{P}_1$* p-simulates *$\mathcal{P}_2$ if there exists a polynomial-time computable function f mapping proofs in $\mathcal{P}_2$ into proofs in $\mathcal{P}_1$, i.e. a function f such that for all $F \in TAUT$ it holds that $\mathcal{P}_2\big(F, \pi\big)$ is true if and only if $\mathcal{P}_1\big(F, f(\pi)\big)$ is true.*

If $\mathcal{P}_1$ p-simulates $\mathcal{P}_2$ it must be the case that $comp_{\mathcal{P}_1} \leq {comp_{\mathcal{P}_2}}^{\mathrm{O}(1)}$. If we have this latter condition but do not know whether the function f exists, we say that $\mathcal{P}_1$ weakly p-simulates *$\mathcal{P}_2$.*

By taking the symmetric closure of the p-simulation relation, we can order proof systems and put them into equivalence classes with respect to their relative complexity. Proof systems belonging to the same equivalence class can be considered as having "essentially" the same complexity (i.e. up to a polynomial).

Definition 2.14 (p-equivalence) *Two propositional proof systems $\mathcal{P}_1$ and $\mathcal{P}_2$ are said to be* p-equivalent *if each proof system p-simulates the other.*

If $\mathcal{P}_1$ p-simulates $\mathcal{P}_2$ but there is no p-simulation in the reverse direction $\mathcal{P}_1$ can be considered to be essentially more efficient than $\mathcal{P}_2$, since $\mathcal{P}_1$ is polynomially bounded for every $F \in \mathit{TAUT}$ with a polynomial-size proof in $\mathcal{P}_2$ but the opposite does not hold.

2.3 Connection to Complexity Theory

As was noted above, the study of proof theory has bearing on questions in computational complexity theory. Before discussing this matter in more detail, let us first recall some basic definitions (for a more detailed treatment the reader is referred to for instance [20]).

An *alphabet* Σ is a finite set of symbols. A *language* L over Σ is any set of finite strings made up of symbols from Σ. The language of all (finite) strings over Σ is denoted Σ^*. All languages L over Σ are subsets of Σ^*.

Given a language L over an alphabet Σ, we can define the *decision problem* D_L of determining which strings $s \in \Sigma^*$ belong to L. A particular instance of the decision problem is a question of the form "is s in L?", to which the answer is "yes" or "no".

The complexity class P is the class of decision problems D solvable in polynomial time. Equivalently, P can be defined as the set of languages L decidable in polynomial time. A language L is *decidable in polynomial time* if there is a polynomial-time computable function $f_L : \Sigma^* \mapsto \{\bot, \top\}$ for which it holds that $f_L(s) = \top$ if and only if $s \in L$ (i.e. if and only if the answer to the corresponding decision problem instance is "yes").

A *verification function* is a binary function $f : \Sigma^* \times \Sigma^* \mapsto \{\bot, \top\}$, where the first argument is an ordinary input string s and the second argument is a string c called a *certificate*. We say that a string s is *verified* by f if there exists a certificate c such that $f(s, c) = \top$. The language L verified by a verification function f is $L = \{s \in \Sigma^* \mid \exists c \in \Sigma^* \text{ such that } f(s, c) = \top\}$. The role of c in the definition above can be understood as providing a verifiable proof of the fact that s is indeed in the language L.

The complexity class NP is the class of languages L (or corresponding decision problems D_L) that can be verified by verification functions f_L computable in polynomial time. More precisely, a language L belongs to NP if and only if there exists a polynomial-time binary function f_L for which it holds that

$$L = \left\{ s \in \Sigma^* \mid \exists c \in \Sigma^* \text{ such that } |c| = |s|^{\mathrm{O}(1)} \text{ and } f(s, c) = \top \right\}. \tag{2.1}$$

If this is the case we say that f_L verifies L in polynomial time. NP stands for "nondeterministic polynomial time", a term that has its origin in an alternate but equivalent definition involving nondeterministic Turing machines that are allowed to guess a certificate and then check it in polynomial time.

Let L_1 and L_2 be languages over alphabets Σ_1 and Σ_2, respectively. We say that L_1 is *polynomial-time reducible* to L_2 and write $L_1 \leq_p L_2$ if there exists a polynomial-time computable function $g : {\Sigma_1}^* \mapsto {\Sigma_2}^*$ such that $s \in L_1$ if and only if $g(s) \in L_2$. This means that if we can test strings for membership in L_2 in time t, we can use g to test strings for membership in L_1 in time polynomial in t. A language L (or decision problem D_L) is NP-*hard* if, for every language $L' \in \mathsf{NP}$, L' is polynomially reducible to L. A language L (or decision problem D_L) is NP-*complete* if it is both NP-hard and in NP.

The complexity class $\mathsf{co\text{-}NP}$ is the set of languages L such that $\overline{L} \in \mathsf{NP}$ (where the language $\overline{L} = \Sigma^* \setminus L$ is called the *complement* of L).

Returning to proof theory, it is well known than the set SAT of satisfiable propositional logic formulas is an NP-complete language. Since a formula F is unsatisfiable if and only if $\neg F$ is a tautology, the set $TAUT$ can be seen to be in $\mathsf{co\text{-}NP}$.

The following theorem by Cook and Reckhow demonstrates the intimate connection between propositional proof systems and one of the central questions of complexity theory, namely whether $\mathsf{P} = \mathsf{NP}$ or not. This connection is one of the main (theoretical) motivations behind the study of propositional proof systems.

Theorem 2.15 (Cook and Reckhow [19])
$\mathsf{NP} = \mathsf{co\text{-}NP}$ *if and only if there exists a p-bounded propositional proof system.*

Proof: ($\Rightarrow$) If $\mathsf{NP} = \mathsf{co\text{-}NP}$, then $TAUT \in \mathsf{NP}$. That is, there is a polynomial-time computable verification function f_{TAUT} such that a formula F is in $TAUT$ if and only if there is a certificate c polynomial in the size of F for which $f_{TAUT}(F, c) = \top$. But then we can pick c as our proof π of F and it is easy to verify that $\mathcal{P} = f_{TAUT}$ is a p-bounded propositional proof system in accordance with definitions 2.10 and 2.12.

($\Leftarrow$) Conversely, assume that there exists a p-bounded propositional proof system $\mathcal{P}$ and let L be a language in NP. To show that $\mathsf{NP} = \mathsf{co\text{-}NP}$ we need to prove that $\overline{L} \in \mathsf{NP}$.

As discussed above, L is polynomial-time reducible to the complement of $TAUT$ in the following sense: there is a polynomial-time function g such that for any string s, it holds that $s \in L$ if and only if $g(s) = F \in SAT$, i.e. if and only if $\neg F \notin TAUT$.

Now let $f_{\overline{L}}$ be the function that on input s computes $g(s) = F$ and accepts if there is a proof π of $\neg F$ in $\mathcal{P}$ (i.e. if $\mathcal{P}(\neg F, \pi) = \top$ for some π). If $s \in \overline{L}$ then $\neg F$ must be a tautology, and since $\mathcal{P}$ is p-bounded there is a polynomial-size certificate $c = \pi$ of this fact. Also, by definition 2.10 the predicate $\mathcal{P}$ is polynomial-time computable. It follows that $f_{\overline{L}}$ verifies $\overline{L}$ in polynomial time in accordance with (2.1), so $\overline{L} \in \mathsf{NP}$ and $\mathsf{NP} = \mathsf{co\text{-}NP}$. $\square$

To see the connection to $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$, suppose that we can show that there are no p-bounded propositional proof systems (i.e. that for all proof systems one can find a family of tautologies which does not have polynomial-size proofs in the system). Then theorem 2.15 says that it must be the case that $\mathsf{P} \neq \mathsf{NP}$ (since $\mathsf{P} = \mathsf{co\text{-}P}$).

2.4 Proof Methods

The definition 2.10 of propositional proof systems is non-constructive. We do not say anything about how the proofs should be found, only that they should be checkable in polynomial time. A *proof method* is a constructive algorithm that on input F generates a proof of F if the formula is valid.

Definition 2.16 (Proof method) *Let $\mathcal{P}$ denote a propositional proof system. A* proof method *(or* proof procedure*) $A_{\mathcal{P}}$ for $\mathcal{P}$ is a deterministic algorithm $A_{\mathcal{P}}$*

that takes as input a formula F and generates a proof π of F in the format specified by the proof system $\mathcal{P}$ (i.e. such that $\mathcal{P}(F,\pi) = \top$) if F is valid and reports that F is falsifiable otherwise.

Given a proof method $A_{\mathcal{P}}$, an important question is how efficient $A_{\mathcal{P}}$ is. The efficiency of $A_{\mathcal{P}}$ is bounded from below by the proof system $\mathcal{P}$ in the sense that the minimal size of a proof of a tautology F in $\mathcal{P}$ places an obvious lower bound on the running time of any proof method for $\mathcal{P}$ on input F. A natural way of measuring the performance of the algorithm $A_{\mathcal{P}}$ is to consider the running time of $A_{\mathcal{P}}$ on input F relative to this minimal proof size. For convenience, we introduce a notation for this measure.

Definition 2.17 *Let $\mathcal{P}$ be a propositional proof system and suppose that F is a tautological propositional logic formula. Then the size of a minimal proof of F in $\mathcal{P}$ is denoted $S_{\mathcal{P}}(\vdash F)$ (or just $S(\vdash F)$ when the proof system is clear from context).*

In this way, we can classify proof systems based on whether there are efficient proof methods for them or not. When choosing between proof systems on which to base, say, an automated theorem prover, ideally we would like to use a proof system which has both small proofs and an algorithm for finding these small proofs efficiently.

As usual, by "efficient" in this context we mean polynomial. We consider a proof method efficient if it finds a proof for a formula F in time polynomial in the size of a smallest proof. A proof system for which such an algorithm exists is called *automatizable*.

Definition 2.18 (Automatizability) *A propositional proof system $\mathcal{P}$ is* automatizable *if there exists a proof method $A_{\mathcal{P}}$ that takes as input a tautology F and outputs a $\mathcal{P}$-proof of F in time polynomial in the size $S_{\mathcal{P}}(\vdash F)$ of a smallest $\mathcal{P}$-proof of F.*

$\mathcal{P}$ is called quasi-automatizable *if the running time of $A_{\mathcal{P}}$ is quasi-polynomial in the size $S_{\mathcal{P}}(\vdash F)$ of a minimal proof, i.e. if*

$$Time\,(A_{\mathcal{P}}(F)) \leq \exp\left((\log S_{\mathcal{P}}(\vdash F))^{\mathrm{O}(1)}\right).$$

2.5 More about Proof Systems and Methods

We continue our study of proof systems and proof methods by giving some examples. As we discuss the examples we also introduce some new terminology and notation.

Perhaps the simplest example imaginable of a propositional proof system is truth tables.

Example 2.1 (Truth tables) Given a formula F, we construct the truth table for F by assigning truth values to the variables in $Vars(F)$ according to all all possible assignments $\alpha \in \{\bot, \top\}^{Vars(F)}$ in turn. For each assignment, we write down the truth values of all variables and subformulas of F in a bottom-up fashion until we reach the formula F itself.

Truth tables are an automatizable proof system, since the algorithm sketched above is obviously linear in the size of the constructed truth table. Unfortunately, this is not very exiting news for the simple reason that a truth table proof is typically exponential in the size $S(F)$ of a formula F. $\diamond$

Despite the simplicity of truth tables, there are examples of formal verification problems in industrial applications where the usage of truth tables seem to be the only practical solution [51].

Of course, in general more advanced proof systems and proof methods are much more efficient. A natural way of designing such proof systems is to define the format of proofs in terms of a *system for natural deduction* [22, 43]. A proof can then be represented as a sequence of propositional logic formulas, where each line in the proof follows from the preceding ones by the rules of the natural deduction system.

Given such a proof format, we can construct a proof system $\mathcal{P}$ as a procedure for checking that each step in the derivation is in accordance with the rules of the deduction system. It is not hard to see that (for reasonable deduction systems) this can be done in time polynomial in the size of the proof, and thus $\mathcal{P}$ is a propositional proof system. (To be more precise, in order to be efficiently checkable, proofs in proof systems based on systems for natural deduction should include annotations for each step in the derivation about how it was derived. Since such annotations will affect proof size only by a constant factor, we will usually ignore this issue.)

For derivations in natural deduction systems, a relevant measure in addition to the size of the derivation is the number of lines in it.

Definition 2.19 (Proof length) *Let $\mathcal{P}$ be a propositional proof system defined in terms of a system for natural deduction.*

The proof length *of a proof π in $\mathcal{P}$, denoted $L(\pi)$, is the number of lines in the proof π. For a tautology F, we let $L_{\mathcal{P}}(\vdash F)$ denote the length of a shortest proof of F in the proof system $\mathcal{P}$.*

Remark 2.20 A note on notation: To avoid ambiguity, we do not use the notation $|\pi|$ or $|F|$ in measures of proofs and formulas. The reason for this is that for instance $|F|$ is used to denote formula size in some articles and the number of clauses of a CNF formula in others, and the notation $|\pi|$ is used to denote proof size in some articles and number of lines of a proof in others.

Instead, we write $S(\pi)$ to denote size (total number of characters) and $L(\pi)$ to denote length (number of lines). This notation is somewhat cumbersome, but has the advantage of being wholly unambiguous.

A characteristic property of most propositional proof systems based on systems for natural deduction is that if there is a proof of a formula F, then there is a proof using only subformulas or negated subformulas of F. This is called the *subformula principle*.

Definition 2.21 (Subformula principle) *A propositional proof system $\mathcal{P}$ or proof π is said to obey the* subformula principle *(or possess the* subformula property*) if the only formulas occurring in the proof of a formula F belong to $Sub(F)$ (i.e. are subformulas of F or negations of such formulas or the constants $\top$ and $\bot$).*

A proof system $\mathcal{P}$ or proof π which respects the subformula principle is said to be analytic.

If a proof system $\mathcal{P}$ uses only subformulas of the formula F to be proved, we can exploit this fact to design proof search algorithms for $\mathcal{P}$ with guaranteed bounds on worst performance. Upper bounds on the running time for such algorithms can be given in terms of the formula size $S(F)$, since the only formulas which need to be considered during the proof search are the subformulas of F.

We conclude this section by giving two examples of (families of) propositional proof systems which can be considered to be part of general knowledge in proof theory.

Example 2.2 (Gentzen systems) In *sequent calculi*, which were introduced by Gentzen [28] and are therefore also known as *Gentzen systems*, proofs consist of expressions on the form

$$F_1, \ldots, F_n \vdash G_1, \ldots, G_m, \tag{2.2}$$

so called *sequents*. In the sequent above, the sequence $F_1, \ldots, F_n$ is called the *antecedent* and $G_1, \ldots, G_m$ is called the *succedent*. They are both referred to as *cedents*.

The intended interpretation of a sequent is that if all of the formulas in the antecedent are true, then one of the formulas in the succedent is true. Thus, the sequent (2.2) is equivalent in meaning to the formula

$$\bigwedge_{i=1}^{n} F_i \to \bigvee_{j=1}^{m} G_j. \tag{2.3}$$

Proofs start from obviously valid sequents of the form $F \vdash F$ (*initial sequents* or *axioms*). Complex sequents are built up by applications of inference rules. For each connective, there are "left" and "right" rules. If we let Γ and Δ denote sequences of formulas, then for instance we have the rules

$$Or_L \ \frac{\Gamma, F \vdash \Delta \quad \Gamma, G \vdash \Delta}{\Gamma, F \vee G \vdash \Delta} \qquad\qquad Or_R \ \frac{\Gamma \vdash \Delta, F, G}{\Gamma \vdash \Delta, F \vee G} \tag{2.4}$$

for disjunction and

$$And_L \ \frac{\Gamma, F, G \vdash \Delta}{\Gamma, F \wedge G \vdash \Delta} \qquad\qquad And_R \ \frac{\Gamma \vdash \Delta, F \quad \Gamma \vdash \Delta, G}{\Gamma \vdash \Delta, F \wedge G} \tag{2.5}$$

for conjunction. Furthermore, there are the two *structural rules*

$$Cut \ \frac{\Gamma, F \vdash \Delta \quad \Gamma \vdash \Delta, F}{\Gamma \vdash \Delta} \tag{2.6}$$

and

$$Thinning \ \frac{\Gamma \vdash \Delta}{\Gamma, F \vdash \Delta, G} \tag{2.7}$$

(thinning is also known as *weakening*). Finally, if the cedents are defined as sequences and not as sets, we need so called *weak structural rules* for purely formal manipulations (such as reordering formulas within a cedent or eliminating duplicates).

Thinning can be used to minimize the number of different sequents in proofs and hence to reduce proof complexity. The cut rule in sequent calculi is not needed for completeness and is thus unnecessary. However, the use of the cut rule can shorten proofs significantly. A proof is said to be *cut-free* if it does not contain any cut inferences.

Gentzen systems are interesting both in their own right and in connection with the dilemma proof system which we will study in chapter 4. We refer to [49] for a discussion of this. $\diamond$

Example 2.3 (Frege systems) A *Frege rule* is a pair $\big(\{F_1,\ldots,F_n\},G\big)$, for propositional logic formulas $F_1,\ldots,F_n,G$ over variables $x_1,\ldots,x_m$, such that the implication $\big(F_1\wedge\ldots\wedge F_n\big)\rightarrow G$ is a tautology. Usually we write the rule as

$$\frac{F_1,\ldots,F_n}{G}. \tag{2.8}$$

In derivations, one uses instances of a Frege rule by substituting arbitrary formulas for the variables $x_1,\ldots,x_m$. If a rule has zero assumptions, it is called an *axiom schema*. A *Frege proof* is a sequence of formulas where each formula follows from previous ones by an application of a Frege rule from a given set.

A *Frege system* is determined by a finite, implicationally complete set of Frege rules based on a functionally complete set of connectives. (A set of rules is *implicationally complete* if whenever $\big(F_1\wedge\ldots\wedge F_n\big)\rightarrow G$ is a tautology, it is also the case that G is derivable from $F_1,\ldots,F_n$.)

Remarkably, although Frege systems are defined without reference to proof size, proof length or any other proof-theoretic measure of efficiency, all Frege systems are polynomially equivalent. Frege systems can be extended by including the *substitution rule*, which allows substitution in derived formulas, or the *extension rule*, which permits the introduction of abbreviations for long formulas. Every two substitution Frege systems and every two extension Frege systems (or *EF-systems*) are p-equivalent. Moreover, Frege system are p-equivalent to Gentzen systems [19]. $\diamond$

We will not discuss Gentzen or Frege systems any further in this thesis. The reader is referred to to the references given at the beginning of this chapter for more details.

2.6 Resolution

It is possible to convert any propositional logic formula F to a formula in conjunctive normal form in such a way that it has only polynomially 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 [56].

The idea in Tseitin's transformation is to introduce a new variable x_P for each subformula $P \doteq Q \circ R$ in F. The formula F is then translated to conjunctive normal form by adding a set of clauses $Cl\big(P\big)$ for each subformula P which enforces that the the truth value of x_P is computed correctly given the truth values of x_Q and x_R. These clauses $Cl\big(P\big)$ are presented in figure 2.1.

$$P \doteq Q \wedge R: \qquad Cl(P) := \begin{aligned}[t] &\left(\overline{x_P} \vee x_Q\right) \\ \wedge\ &\left(\overline{x_P} \vee x_R\right) \\ \wedge\ &\left(x_P \vee \overline{x_Q} \vee \overline{x_R}\right) \end{aligned}$$

$$P \doteq Q \vee R: \qquad Cl(P) := \begin{aligned}[t] &\left(\overline{x_P} \vee x_Q \vee x_R\right) \\ \wedge\ &\left(x_P \vee \overline{x_Q}\right) \\ \wedge\ &\left(x_P \vee \overline{x_R}\right) \end{aligned}$$

$$P \doteq Q \rightarrow R: \qquad Cl(P) := \begin{aligned}[t] &\left(\overline{x_P} \vee \overline{x_Q} \vee x_R\right) \\ \wedge\ &\left(x_P \vee x_Q\right) \\ \wedge\ &\left(x_P \vee \overline{x_R}\right) \end{aligned}$$

$$P \doteq Q \leftrightarrow R: \qquad Cl(P) := \begin{aligned}[t] &\left(\overline{x_P} \vee \overline{x_Q} \vee x_R\right) \\ \wedge\ &\left(\overline{x_P} \vee x_Q \vee \overline{x_R}\right) \\ \wedge\ &\left(x_P \vee \overline{x_Q} \vee \overline{x_R}\right) \\ \wedge\ &\left(x_P \vee x_Q \vee x_R\right) \end{aligned}$$

Figure 2.1: Tseitin's transformation to CNF clauses.

(Of course, if Q or R in $P \doteq Q \circ R$ is a negated subformula, say $Q \doteq \neg Q'$, x_Q is replaced by $\overline{x_{Q'}}$.) Finally, a unit clause $\overline{x_F}$ 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.

One such system which is widely used as the basis for different proof search algorithms is *resolution* [47]. Resolution can be viewed as a very specialized form of a Frege system that can only manipulate clauses and has only one inference rule, namely the rule that from the clauses $B \vee x$ and $C \vee \neg x$ derives $B \vee C$. This rule can be seen to be a special form of the cut rule (2.6). The contradictory formula to be derived is simply an empty clause. Although resolution is a very simple proof system, it has been popular as a basis for proof search algorithms in for instance formal verification and artificial intelligence research.

2.6.1 Definitions and Elementary Results

We now describe resolution in more detail. First, we make exact our choice of terminology by giving some elementary definitions.

Definition 2.22 *A* literal *over a propositional logic variable x is either x itself or its negation $\neg x$ (alternatively denoted $\overline{x}$). In some contexts it will be convenient to use the notation x^1 for x and x^0 for $\overline{x}$.*

A clause *is a disjunction of literals. A clause containing exactly k literals is called a* k-clause. *We say that a clause C is* ordinary *if there is no variable x such that both x and $\overline{x}$ are literals in C.*

A CNF formula *is a conjunction of clauses. A* k-CNF formula *is a CNF formula consisting of k-clauses.*

Remark 2.23 It is a somewhat unfortunate fact that the term "k-CNF formula" is used to mean a formula with *exactly* k literals in each clause in some articles and a formula with *at most* k literals in each clause in others.

Probably, it would be better to use (for instance) the term "Ek-CNF formula" for a CNF formula with exactly k literals in each clause and let k-CNF formulas be the more general class of CNF formulas with clauses of at most k literals. Since the definition of k-CNF formulas given in definition 2.22 seems to be the one used in the literature in connection with bounds for resolution refutations, however, we have chosen to use it in this thesis.

We use the lower-case letters a, b, c and d, with or without indices, to denote literals. Also, we adopt the notational convention that $\overline{a} := x$ if $a = \overline{x}$. The upper-case letters A, B, C and D are used to denote clauses. Normally, only ordinary clauses will be of interest.

In the context of CNF formulas F we always let n denote the number of variables in F and m the number of clauses in F. One important measure for random k-CNF formulas is the quotient of the number of clauses and the number of variables.

Definition 2.24 (Density) *For a k-CNF formula F with m clauses over n variables, we say that $\Delta := m/n$ is the* density *of F.*

The intuition behind this definition is that a dense formula has many clauses over comparatively few variables, and so is likely to be "overdetermined" (that is, unsatisfiable). In contrast, a sparse k-CNF formula has few clauses restricting the possible truth values of the variables, and is therefore likely to be satisfiable. We will return to the concept of density in section 3.3.

In some articles, alternate definitions are used of a clause C as a set of literals and a CNF formula F as a set of clauses. Although possibly somewhat counter-intuitive, these definitions have a number of advantages. For instance, it becomes very natural to use set-theoretic notation like $a \in B$ or $C \in F$ to mean that the literal a occurs in B or that C is a clause in F, respectively. More significantly, we automatically get that $\vee$ is commutative and associative, so that $a \vee b \vee c$ and $c \vee b \vee a$ are two ways of writing the same clause, and that this clause is identical to $a \vee b \vee c \vee b \vee a$, since clauses are sets and duplicates are thus automatically removed.

To make use of these advantages, we will sometimes allow ourselves to switch between definition 2.22 and the definition sketched in the last paragraph. More specifically, we will always assume that the order between the literals in a clause is insignificant and that there are no duplicates of literals or clauses.

Our choice of notation concerning CNF formulas is described in the following definition.

Definition 2.25 (Notation for CNF formulas) *For CNF formulas F, F' and a clause C we let $C \in F$ denote that C is one of the clauses of F. We use*

$F' \subseteq F$ to mean that all clauses in F' are also found in F, and $F \setminus F'$ denotes the CNF formula containing all clauses in F that are not clauses of F'. $C \setminus \{a\}$ denotes the clause C with any occurrences of the literal a deleted.

The set of variables in a clause C is denoted $Vars(C)$. $Lit(C)$ denotes the set of literals in C. We extend these definitions to CNF formulas F in the natural way by defining $Vars(F) := \bigcup_{C \in F} Vars(C)$ and $Lit(F) := \bigcup_{C \in F} Lit(C)$.

Thus, in this thesis the fact that x is a variable in C is expressed by the notation $x \in Vars(C)$ and we write $a \in Lit(F)$ to express that the literal a occurs in the formula F. While this notation might seem to be somewhat awkward, just as in remark 2.20 we feel that it has the advantage of eliminating some possible sources of ambiguity (for example, it is not clear whether $x \in C$ should mean that x is one of the *variables* in C or that the *literal* x occurs in C).

We are now ready to define more formally what a resolution derivation is.

Definition 2.26 (Resolution) *A* resolution derivation *of a clause A from a CNF formula F is a sequence of clauses $\pi = \{D_1, \ldots, D_s\}$ such that $D_s = A$ and each line D_i, $1 \leq i \leq s$, is either one of the clauses in F or is derived from clauses D_j, D_k in π (with $j, k < i$) by the* resolution rule

$$\frac{B \vee x \quad C \vee \overline{x}}{B \vee C} \tag{2.9}$$

(where the variable x and the clauses B, C are arbitrary). We refer to (2.9) *as* resolution on the variable x *and $B \vee C$ is called the* resolvent *of the clauses $B \vee x$ and $C \vee \overline{x}$ on x.*

A resolution refutation *of a CNF formula F is a resolution derivation of the empty clause (the clause with no literals), denoted* 0 *or* Λ, *from F.*

A resolution derivation is tree-like *if any clause in the derivation is used at most once as a premise in an application of the resolution rule.*

Every truth assignment satisfying both $B \vee x$ and $C \vee \overline{x}$ obviously must satisfy the resolvent $B \vee C$ as well. By straightforward induction, any truth assignment satisfying F must satisfy all clauses D_i in a resolution derivation from F. Since the last step in a refutation of F by resolution must be $\frac{x \quad \overline{x}}{0}$ for some variable x, a resolution refutation of F as defined above clearly is a proof of the fact that F is unsatisfiable. If we view a resolution refutation as a directed acyclic graph (DAG) with clauses as nodes and edges from the premises to the conclusion of each resolution step, a resolution refutation is tree-like if its DAG is a tree. Note that resolution is not an analytic proof system.

Remark 2.27 In some articles the (arguably somewhat unfortunate) phrases "resolution proof for F" or "resolution proof of F" is used to mean a resolution refutation of F. We will avoid this usage, but will otherwise consider the terms "resolution refutation" and "resolution proof" to be synonymous.

Definition 2.19 of proof length extends naturally to resolution in the following manner.

Definition 2.28 (Length) *The* length *$L(F)$ of a CNF formula F is the number of clauses in F. In the same way, the length $L(\pi)$ of a resolution derivation π is the number of clauses (counted with repetitions) in π.*

The length of refuting F by resolution, denoted $L_{\mathcal{R}}(F \vdash \bot)$ (or just $L(F \vdash \bot)$ without index when the proof system is clear from context), is defined to be $L_{\mathcal{R}}(F \vdash \bot) := \min_{\pi} \{L(\pi)\}$, where the minimum is taken over all resolution refutations π of F.

In an analogous fashion, the length of refuting F by tree-like resolution is $L_{\mathcal{T}}(F \vdash \bot) := \min_{\pi_{\mathcal{T}}} \{L(\pi_{\mathcal{T}})\}$, where the minimum is taken over all tree-like resolution refutations $\pi_{\mathcal{T}}$ of F.

In addition to size and length, an important measure on a resolution derivation π is the *width* of π, i.e. the maximum number of literals in any clause in π.

Definition 2.29 (Width) *The* width $W(C)$ *of a clause C is the number of literals appearing in it. The width of a formula (or set of clauses) F is the maximal width of a clause in the formula (or set), $W(F) := \max_{C \in F} \{W(C)\}$, and the width of a resolution derivation π is $W(\pi) := \max_{D \in \pi} \{W(D)\}$.*

The width of deriving a clause A from the formula F by resolution, denoted $W_{\mathcal{R}}(F \vdash A)$ (or just $W(F \vdash A)$ when the meaning is clear from context), is defined as $W_{\mathcal{R}}(F \vdash A) := \min_{\pi} \{W(\pi)\}$, where the minimum is taken over all resolution derivation π of A from F.

We use the notation $F \vdash_w A$ to mean that A can be derived from F in width less than or equal to w.

We will mainly be interested in the width $W_{\mathcal{R}}(F \vdash \bot)$ of refuting a CNF formula F by resolution. (Note that the width measure is independent of whether we use tree-like or general resolution, i.e. $W_{\mathcal{R}}(F \vdash A) = W_{\mathcal{T}}(F \vdash A)$.)

When proving theorems about resolution refutation, it is sometimes convenient to add a derivation rule for thinning or weakening, the *weakening rule*

$$\frac{B}{B \vee C} \tag{2.10}$$

(for arbitrary clauses B, C). It is a routine matter to show that the use of thinning in a resolution refutation can be eliminated to produce a proof that employs only the resolution rule.

Proposition 2.30 (Elimination of weakening rule)
Suppose that π is a refutation of a CNF formula F using the resolution rule (2.9) *and the weakening rule* (2.10). *Then π can be transformed to a resolution refutation π' of F using only the resolution rule without increasing the length, width or size of the proof.*

Sketch of proof: This is an easy induction over the refutation π. It is obvious from the construction in the induction proof that the length, width and size of the resolution refutation can only decrease as a result of the transformation. □

In view of this proposition, we will allow ourselves to use the weakening rule to simplify the proofs for some of the theorems about resolution presented in this thesis. We prove most of our theorems using resolution extended with the weakening rule, tacitly assuming that the transformation described by proposition 2.30 is applied to all refutations in order to reduce them to resolution proofs in the strict sense of definition 2.26.

One important technique for proving bounds on resolution proofs is the application of randomly chosen *restrictions* on formulas and derivations. Intuitively, restricting a formula means assigning truth values to a set of variables occurring in the formula and simplifying the result.

Definition 2.31 (Restriction) *For C a clause, x a variable and $\nu \in \{0,1\}$ a truth value, the* 1-restriction *of x to ν in C is*

$$C|_{x=\nu} := \begin{cases} C & \text{if } x \notin \mathit{Vars}(C), \\ 1 & \text{if } x^{\nu} \in \mathit{Lit}(C), \\ C \setminus \{x^{1-\nu}\} & \text{if } x^{1-\nu} \in \mathit{Lit}(C). \end{cases}$$

We extend the definition of 1-restrictions by defining $F|_{x=\nu} := \bigwedge_{C \in F} C|_{x=\nu}$ for CNF formulas F and $\pi|_{x=\nu} := \{D_1|_{x=\nu}, \ldots, D_s|_{x=\nu}\}$ for resolution derivations $\pi = \{D_1, \ldots, D_s\}$.

A partial assignment *or* restriction *ρ is a partial function $\rho : V \mapsto \{0,1\}$, where V is a set of boolean variables of a clause or formula.*

For a clause C, let $\mathcal{D} \subseteq \mathit{Vars}(C)$ be the domain of ρ, i.e. the set of variables for which the function is defined, and let $x_1, \ldots, x_t$ be some ordering of the variables in $\mathcal{D}$. Then the ρ-restriction *of C is defined to be*

$$C|_{\rho} := (\ldots(C|_{x_1=\rho(x_1)})|\ldots)|_{x_t=\rho(x_t)}.$$

In analogy with the definition of 1-restrictions, we extend the definition of general restrictions to formulas and derivations by stipulating $F|_{\rho} := \bigwedge_{C \in F} C|_{\rho}$ and $\pi|_{\rho} := \{D_1|_{\rho}, \ldots, D_s|_{\rho}\}$ if $\pi = \{D_1, \ldots, D_s\}$.

In the definition above, 1 denotes the trivially true clause. The result of resolving any clause C with 1 is defined to be C. In the following, we assume without loss of generality that $\pi|_{\rho}$ does not contain any clauses 1 (by removing all such clauses from the derivation if need be).

Before we proceed any further we note that restriction is well-defined. Also, we note that the property of being a resolution refutation is preserved under restrictions.

Proposition 2.32
$C|_{\rho}$, $F|_{\rho}$ and $\pi|_{\rho}$ as described in definition 2.31 are well-defined.

Proposition 2.33
If π is a resolution refutation of F and ρ is a restriction on $\mathit{Vars}(F)$, then $\pi|_{\rho}$ is a refutation of $F|_{\rho}$ using the resolution and (possibly) weakening rules.

Sketch of proof [of propositions 2.32 and 2.33]: It is enough to prove proposition 2.32 for a restriction over a clause $C|_{\rho}$, which is straightforward. Proposition 2.33 can be shown for 1-restrictions by an easy induction over the resolution refutation, from which the general case follows. □

Note that by proposition 2.30, $\pi|_{\rho}$ can be transformed to a refutation of (at most) the same size, length and width using the resolution rule only.

Note also that the definition of restriction above can be extended to sets of clauses in the same way that it was extended to formulas and proofs. When convenient, we will tacitly assume such extensions in the following.

2.6.2 Proof Methods for Resolution

We conclude our introduction to resolution with a short presentation of two proof methods. The input to the algorithms is a (supposedly contradictory) CNF formula F on n variables. Both algorithms reduce the problem of refuting F to a problem on $n-1$ variables, but in rather different ways.

Example 2.4 (Davis-Putnam) Given a CNF formula F on n variables, pick a variable $x \in F$ and eliminate it by replacing the set of clauses containing x with all their possible resolvents. The algorithm which performs this reduction repeatedly for $n, n-1, \ldots, 1$ until finally the empty clause is derived (or the formula is found to be satisfiable) is due to Davis and Putnam [24], and is consequently known as the *Davis-Putnam procedure*. The resolution proof system restricted to proofs produced by the Davis-Putnam procedure is called *Davis-Putnam resolution* (or *DP-resolution* for short).

Although the Davis-Putnam procedure might seem to be a rather general proof search algorithm for resolution, the restriction that the variables be eliminated one at a time is a severe one. DP-resolution has been shown to be exponentially weaker than general resolution [10]. $\diamond$

One obvious disadvantage of Davis-Putnam is the large memory space it might use. Saving all possible resolvents might require space that is exponential in n. In the next example we present a proof search algorithm which is potentially much weaker than Davis-Putnam with respect to proof length, but is more efficient with respect to space and therefore is the algorithm preferred in practice.

Example 2.5 (DLL) A simple scheme for a family of algorithms for refuting a contradictory CNF formula F on n variables is as follows: If the empty clause 0 is in F, report that F in unsatisfiable and halt. Otherwise, pick a variable $x \in F$ and recursively try to refute $F|_{x=0}$ and $F|_{x=1}$. This family of proof methods was introduced by Davis, Logemann and Loveland [23], and such algorithms are therefore known as *DLL procedures*.

DLL procedures have recursion depth bounded by the number of variables n. The space required is thus at most linear in the input size, so DLL procedures are very space-efficient. Their main flaw is that the execution corresponds to a decision tree for the problem of finding a clause falsified by a given truth value assignment to the variables in F. Such a decision tree is isomorphic to a tree-like resolution refutation of F. In other words, all DLL algorithms produce tree-like proofs. This means that DLL procedures can be time costly on inputs that do not have short tree-like refutations, and it has been shown that tree-like resolution is exponentially weaker than general resolution [6] (see also section 3.2) and even DP-resolution [11].

A particular DLL algorithm is specified by a *splitting rule*, which is the subroutine that for each recursively constructed formula determines the next variable to split over and which truth value to try first. Different splitting rules may result in vastly different running times.

If a formula F contains a *unit clause* (i.e. a clause consisting of a single literal x or $\overline{x}$), setting this unit clause to *FALSE* falsifies the whole formula F. A simple principle for any splitting rule, therefore, is to set the literal in the unit clause to *FALSE*, immediately terminate the *FALSE*-branch (since F was falsified), set the literal to *TRUE* and then continue. A splitting rule which always chooses unit clauses when possible is said to obey *unit resolution* or *unit propagation*. Virtually all splitting rules considered in the literature obey unit propagation, and any splitting rule can be modified to obey unit propagation at the cost of a factor $\mathrm{O}(n)$ [3] (where as before n is the number of variables).

The simplest such splitting rule is to fix an ordering $x_1, \ldots, x_n$ of the variables in F. For all subformulas F' that arise at some point in the execution of the algorithm, if there is a unit clause the splitting rule chooses the variable in the first such clause. Otherwise the first unfixed variable with respect to the ordering is selected. The DLL procedure obtained from this splitting rule is called *fixed-order DLL*. $\diamond$

Remark 2.34 Confusingly enough, some authors use the label "Davis-Putnam" to refer to DLL procedures. Since the difference between Davis-Putnam and DLL procedures is significant (in the worst case exponential), we choose to make the distinction between the two methods clear by adopting the terminology presented above.

For an example of another approach to proof search in the resolution proof system, see the algorithm presented in section 3.1.3.

Chapter 3

Bounds for Resolution

The resolution proof system introduced in section 2.6 is one of the most well-studied models in proof theory [5]. In this chapter we discuss resolution in greater detail and give some current results for bounds on proof length, which are used as a basis for the theorems proved in chapter 6. The main references for the material presented below are Ben-Sasson and Wigderson [7] and Beame et al. [3] (based on their earlier article [2]).

3.1 Relating Width and Length of Proofs

If the minimal resolution refutation π of a CNF formula F is long, it seems rather natural to assume that π contains clauses with many literals. Conversely, short proofs can be expected to be narrow as well.

In this section, we make this intuition precise by relating proof width to proof length in tree-like and general resolution. The two key results presented are:

- If a contradictory CNF formula F has a tree refutation of length L_T, then it has a refutation of maximal width $\log_2 L_T$.

- If a contradictory CNF formula F has a general resolution refutation of length L, then it has a refutation of maximal width $\mathrm{O}\left(\sqrt{n \log L}\right)$ (where n is the number of variables in F).

Thus, if we want to give a lower bound on the length of a resolution proof, we can concentrate on finding a lower bound for the *width* of a proof. In the other direction, the width-length relations rather naturally suggest a simple dynamic programming procedure for automated theorem proving, namely one that searches for small width proofs (see section 3.1.3).

The results presented in this section are from [7], and our presentation follows that article rather closely. The main difference is that [7] uses the term "size" instead of "length" for the number of lines in a proof. We reserve the word size to mean the total number of symbols in a proof in order to be able to reason about p-simulations without too much terminological confusion. As a consequence of this, and also in order to achieve notational consistency throughout this thesis, we use a slightly different notation from that of [7]. Moreover, minor modifications have been made in a couple of definitions to eliminate some gaps of a purely technical nature in the proofs.

3.1.1 The Length-Width Relations

Stated informally, theorems 3.3 and 3.5 below say that if a CNF formula F has a *short* resolution refutation, then it has a refutation with *small width*.

We start by proving two technical lemmas (where $F|_{x=\nu}$ denotes a restriction as defined in definition 2.31 on page 23).

Lemma 3.1
For $\nu \in \{0,1\}$, if $F|_{x=\nu} \vdash_w A$ then $F \vdash_{w+1} A \vee x^{1-\nu}$ (possibly by use of the weakening rule).

Proof: Consider a derivation $\pi = \{D_1, \ldots, D_s\}$ of A from $F|_{x=\nu}$ in width $W(\pi) \leq w$. Add the literal $x^{1-\nu}$ to all clauses in π. We claim that this gives a legal derivation π' of $A \vee x^{1-\nu}$ from F.

Obviously, $W(\pi') \leq w + 1$, and the last line in π' is $A \vee x^{1-\nu}$. To prove that π' is a resolution derivation, we need to show that each $D_i \vee x^{1-\nu} \in \pi'$ can be derived according to the resolution rule (2.9) and the weakening rule (2.10).

Let $F_{x^{1-\nu}} := \left\{C \in F \mid x^{1-\nu} \in \mathit{Lit}(C)\right\}$ be the set of all clauses of F containing the literal $x^{1-\nu}$. We get three cases:

1. $D_i \in F_{x^{1-\nu}}|_{x=\nu}$: Then $D_i \vee x^{1-\nu} \in F$, so this line in the derivation is certainly legal.
2. $D_i \in F|_{x=\nu} \setminus F_{x^{1-\nu}}|_{x=\nu}$: This means that $D_i \in F$, so $D_i \vee x^{1-\nu}$ can be derived by weakening.
3. D_i is derived from $D_j, D_k \in \pi$ by the resolution rule $\frac{D_j \quad D_k}{D_i}$: In this case, by induction $D_i \vee x^{1-\nu}$ can be derived by the resolution rule from earlier clauses $D_j \vee x^{1-\nu}, D_k \vee x^{1-\nu} \in \pi'$.

The lemma follows. □

Lemma 3.2
For $\nu \in \{0,1\}$, let $F_{x^\nu} := \{C \in F \mid x^\nu \in \mathit{Lit}(C)\}$ denote the set of all clauses in F containing the literal x^ν.

If $F|_{x=\nu} \vdash_{w-1} 0$ and $F|_{x=1-\nu} \vdash_w 0$ then $W(F \vdash \perp) \leq \max\{w, W(F_{x^\nu})\}$.

Proof: Suppose that $F|_{x=\nu} \vdash_{w-1} 0$ and $F|_{x=1-\nu} \vdash_w 0$.

By lemma 3.1, $F \vdash_w x^{1-\nu}$. Having derived the clause $x^{1-\nu}$, we resolve it with the clauses in F_{x^ν} one by one to yield all clauses in $F|_{x=1-\nu}$. The width of this part of the derivation is $W(F_{x^\nu})$. Finally, we derive a contradiction from the clauses in $F|_{x=1-\nu}$, which by assumption can be done in width at most w.

Since according to proposition 2.30 all applications of the weakening rule can be eliminated without increasing the width of the proof, we are finished. (In the following, we will no longer explicitly eliminate uses of weakening, but implicitly assume the application of proposition 2.30 when needed.) □

Lemma 3.2 is our main tool in the proofs of the upper bounds for width given in the theorems below.

Theorem 3.3
For tree-like resolution, the width of refuting a CNF formula F is bounded from above by

$$W(F \vdash \bot) \leq W(F) + \log_2 L_{\mathcal{T}}(F \vdash \bot).$$

Proof: We prove by induction over b, and for each b by nested induction over the number of variables n, that if $L_{\mathcal{T}}(F \vdash \bot) \leq 2^b$ then $W(F \vdash \bot) \leq W(F) + b$.

If $b = 0$ then $0 \in F$ and we are done.

For the induction step over b, suppose that for all $b' < b$ it holds that if $L_{\mathcal{T}}(F \vdash \bot) \leq 2^{b'}$ then $W(F \vdash \bot) \leq W(F) + b'$. Suppose furthermore (for the induction over n) that for all contradictory CNF formulas F on $n' < n$ variables it holds that $W(F \vdash \bot) \leq W(F) + b$ if $L_{\mathcal{T}}(F \vdash \bot) \leq 2^b$. (The induction base $n = 1$ for formulas on only one variable is trivial for all b).

Let F be a contradictory CNF formula on n variables and let π be a tree refutation of F of minimal length $L_T \leq 2^b$. The last step in the derivation is $\frac{x \quad \overline{x}}{0}$ for some variable x, where x and $\overline{x}$ have been derived by tree derivations T_x and $T_{\overline{x}}$ respectively and $L_T = L(T_x) + L(T_{\overline{x}}) + 1$. By proposition 2.33, $T_x|_{x=0}$ and $T_{\overline{x}}|_{x=1}$ are tree refutation of $F|_{x=0}$ and $F|_{x=1}$ of lengths at most $L(T_x)$ and $L(T_{\overline{x}})$ respectively.

Since $L_T \leq 2^b$, one of T_x and $T_{\overline{x}}$ must have length less than 2^{b-1}. Suppose without loss of generality that $L(T_x) \leq 2^{b-1}$. Then by induction over b it follows that $W(F|_{x=0} \vdash \bot) \leq W(F) + b - 1$. Furthermore, $T_{\overline{x}}|_{x=1}$ is a tree refutation of length $L(T_{\overline{x}}) \leq 2^b$ of a formula $F|_{x=1}$ with (at most) $n-1$ variables, so by induction over n we have $W(F|_{x=1} \vdash \bot) \leq W(F) + b$. By lemma 3.2, $W(F \vdash \bot) \leq W(F) + b$. The theorem follows by induction. $\square$

Corollary 3.4
For tree-like resolution, the length of refuting a CNF formula F is bounded from below by

$$L_{\mathcal{T}}(F \vdash \bot) \geq 2^{(W(F \vdash \bot) - W(F))}.$$

Theorem 3.5
For general resolution, the width of refuting a CNF formula F is bounded from above by

$$W(F \vdash \bot) \leq W(F) + \mathrm{O}\left(\sqrt{n \log L_{\mathcal{R}}(F \vdash \bot)}\right)$$

(where n is the number of variables in F).

Proof: Fix a contradictory CNF formula F on n variables and let π be a refutation of F of minimal length L.

If $L = 1$ then $0 \in F$ and we are done, so suppose $L > 1$. Set $w := W(F)$, $d := \lceil \sqrt{2n \ln L}\, \rceil$ and $a := \left(1 - \frac{d}{2n}\right)^{-1}$ (where we have $a > 1$ since $L_{\mathcal{R}}(F \vdash \bot) \leq 2^n \cdot 2^n < e^{2n}$).

Let F' be a contradictory CNF formula with $\mathit{Vars}(F') \subseteq \mathit{Vars}(F)$ and $W(F') \leq W(F)$, and let π' be a refutation of F'. Let π'^* denote the set of *fat* clauses of π' of width strictly greater than d. We claim that if the number of fat clauses $|\pi'^*| < a^b$ then $W(F' \vdash \bot) \leq w + d + b$.

Now F satisfies the conditions on F' and clearly $|\pi^*| < L \leq a^b$ for $b = \log_a L$. Furthermore, we claim that $\log_a L = \mathrm{O}(\sqrt{n \ln L})$. Combining the two claims, we get the required inequality $W(F \vdash \bot) \leq W(F) + \mathrm{O}(\sqrt{n \ln L})$, which proves the theorem.

It remains to prove the claims.

For the first claim, we again use induction over b, and for each b nested induction over the number of variables n.

If $b = 0$ then π' has no fat clauses and trivially $W(F' \vdash \perp) \leq w + d$.

In the induction step, just as in the proof of theorem 3.3 we want to apply a smart restriction and then appeal to lemma 3.2. The key here is to choose a restriction that minimizes the number of fat clauses.

There are $2n$ literals in F', so by the pigeonhole principle there is some literal, say (without loss of generality) x^1, appearing in more than $\frac{d}{2n}\left|\pi'^*\right|$ fat clauses. Restricting $x = 1$ removes all clauses where x^1 appears and yields a refutation $\pi'|_{x=1}$ of $F'|_{x=1}$ having at most $\left(1 - \frac{d}{2n}\right)\left|\pi'^*\right| < a^{b-1}$ fat clauses. By the induction hypothesis for b we have $F'|_{x=1} \vdash_{(w+d+b-1)} 0$. $F'|_{x=0}$ has (at most) $n - 1$ variables and for the fat clauses we have $\left|\pi'|_{x=0}{}^*\right| < a^b$, so by induction over n we get $F'|_{x=0} \vdash_{(w+d+b)} 0$ (again, the base case $n = 1$ is trivial regardless of b). By lemma 3.2, $F' \vdash_{(w+d+b)} 0$ and the claim follows.

As to the second claim, $\log_a L = \ln L / \ln a$. Now $\ln a = -\ln\left(1 - \frac{d}{2n}\right) \geq \frac{d}{2n} \geq \sqrt{\ln L / 2n}$ (where we use the inequality $\ln(1 + x) \leq x$). Putting this together, we get $\log_a L = \mathrm{O}\left(\sqrt{n \ln L}\right)$ as claimed. □

Corollary 3.6
For general resolution, the length of refuting a CNF formula F is bounded from below by

$$L_{\mathcal{R}}(F \vdash \perp) \geq \exp\left(\Omega\left(\frac{(W(F \vdash \perp) - W(F))^2}{n}\right)\right).$$

3.1.2 A Proof Strategy for Lower Bounds

We now turn to the question of how the results in section 3.1.1 can be used to obtain bounds on proof length by proving lower bounds on the width of refuting a CNF formula F. The strategy described in this section consists of four steps:

1. Define a complexity measure $\mu : \{\text{Clauses}\} \mapsto \mathbb{N}^+$ such that $\mu(C) = 1$ for all $C \in F$.
2. Prove that $\mu(0)$ must be large (where 0 is the empty clause concluding any resolution refutation).
3. Infer that in every refutation π of F there must be a clause D with *medium-sized* complexity measure $\mu(D)$.
4. Prove that if the measure $\mu(D)$ of a clause $D \in \pi$ is medium then the width $W(D)$ is *large*.

Once we have showed a lower bound on the width, we appeal to corollary 3.6 to get a lower bound on the length of a resolution refutation (or corollary 3.4 for a lower bound for tree-like resolution).

We now formalize and explain this strategy. First we define a measure that takes care of steps 1–3 above.

Definition 3.7 (Complexity measure) *Let Γ be an unsatisfiable set of boolean functions, i.e. $\Gamma \models 0$ (but such that the function identically false for* *all* $\alpha \in \{0,1\}^{Vars(\Gamma)}$ *is not in* Γ*) and let* C *be a clause. The* complexity measure *of* C *with respect to* Γ *, denoted* $\mu_\Gamma(C)$*, is defined as*

$$\mu_\Gamma(C) := \min\{|\Gamma'| : \Gamma' \subseteq \Gamma,\ \Gamma' \models C\}.$$

It follows immediately from the definition that μ_Γ is a sub-additive complexity measure with respect to resolution steps. We state this as a lemma.

Lemma 3.8
Let Γ *be an unsatisfiable set of boolean functions and suppose that* $\frac{B \quad C}{D}$ *is an application of the resolution rule. Then* $\mu_\Gamma(D) \leq \mu_\Gamma(B) + \mu_\Gamma(C)$*.*

In order to satisfy step 1 of our strategy, we want clauses $C \in F$ to have a small measure $\mu_\Gamma(C)$.

Definition 3.9 (Compatibility) *Let* Γ *be an unsatisfiable set of boolean functions and* F *a CNF formula. We say that* Γ *is* compatible *with* F *if* $\mu_\Gamma(C) = 1$ *for all clauses* $C \in F$*.*

Only compatible Γ will be used for our complexity measures.

Note that step 2 puts another requirement on Γ, namely that no small subset of it should be unsatisfiable. Intuitively, this means that the boolean functions in Γ should come from a “hard” contradiction.

We now turn to step 3 in our strategy. Given any resolution refutation π of a formula F, $\mu_\Gamma(0)$ denotes the complexity of the final empty clause of the refutation π with respect to Γ. If we use a compatible set Γ, by the sub-additivity of μ_Γ we have that for every interval $[c, 2c] \subseteq [1, \mu_\Gamma(0)]$ there must be a $D \in \pi$ with $\mu_\Gamma(D) \in [c, 2c]$.

Lemma 3.10
Suppose that Γ *is a set of boolean functions compatible with an unsatisfiable CNF formula* F *and let* $\gamma \in (2, \mu_\Gamma(0))$*. Then for every resolution refutation* π *of* F *there is a clause* $D \in \pi$ *with*

$$\frac{\mu_\Gamma(0)}{\gamma} \leq \mu_\Gamma(D) \leq 2\frac{\mu_\Gamma(0)}{\gamma}.$$

Proof: Let $S = \{B \in \pi \mid \mu_\Gamma(B) \geq \mu_\Gamma(0)/\gamma\}$. $S \neq \emptyset$ since $0 \in S$. Consider the first clause D in the proof π which is a member of S. $D \notin F$ since by the compatibility of Γ it holds that $\mu_\Gamma(C) = 1$ for $C \in F$. Hence D is derived by the resolution rule from clauses $B_1, B_2 \in \pi \setminus S$ with $\mu_\Gamma(B_i) < \mu_\Gamma(0)/\gamma$. By sub-additivity, $\mu_\Gamma(D) < 2\,\mu_\Gamma(0)/\gamma$. □

Remark 3.11 We deviate from [7] by using the constant γ in lemma 3.10 above (and in the following) to “move” the interval where we want our “medium complex” clause D to be. γ is introduced for purely technical reasons to fill in a small gap in the proofs in [7]. Typical values for γ could be $\gamma = 3$ or $\gamma = 2 + \epsilon$ for a small $\epsilon > 0$.

The rest of this section is devoted to step 4 in our strategy. We prove that medium-sized complexity measure implies large width by relating both concepts to the *expansion properties* of sets of *sensitive* functions compatible with F.

Definition 3.12 (Sensitivity) *A boolean function f is* sensitive *if any two distinct falsifying assignments $\alpha, \beta \in f^{-1}(0)$ have Hamming distance greater than 1 (i.e. differ in at least two variables).*

Definition 3.13 (Critical assignment) *For Γ a set of boolean functions and $f \in \Gamma$, $\alpha \in \{0,1\}^{Vars(f)}$ is a* critical assignment *for f if*

$$\alpha(g) = \begin{cases} 0 & \text{if } g = f, \\ 1 & \text{if } g \in \Gamma,\ g \neq f. \end{cases}$$

Definition 3.14 (Boundary) *For truth value assignments α, β, we say that β is the result of* flipping x in α *(or* flipping α on x*) if*

$$\beta(y) = \begin{cases} 1 - \alpha(y) & \text{if } y = x, \\ \alpha(y) & \text{otherwise.} \end{cases}$$

A boolean function f is dependent *on a variable x if there is an assignment $\alpha \in \{0,1\}^{Vars(f)}$ such that $\alpha(f) = 0$ but flipping x in α satisfies f.*

The boundary *of a set of boolean functions Γ, denoted $\partial\Gamma$, is the set of variables $x \in Vars(\Gamma)$ such that there is a* unique *function $f \in \Gamma$ dependent on x.*

The next lemma is an immediate consequence of the definitions above.

Lemma 3.15
Suppose that Γ is a set of boolean functions, $f \in \Gamma$ is a sensitive function, α is a critical assignment for f and $x \in Vars(f) \cap \partial\Gamma$. Then flipping x in α yields an assignment β which satisfies all $g \in \Gamma$.

Loosely put, we define the *expansion* of Γ to be the size of the minimal boundary of all medium size subformulas of Γ.

Definition 3.16 (γ-expansion) *Let Γ be an unsatisfiable set of boolean functions, and let $\gamma \in \big(2, \mu_\Gamma(0)\big)$. The* γ-expansion *of Γ, denoted $e_\gamma(\Gamma)$, is defined as*

$$e_\gamma(\Gamma) := \min\left\{ |\partial\Gamma'| : \Gamma' \subseteq \Gamma,\ \frac{\mu_\Gamma(0)}{\gamma} \leq |\Gamma'| \leq 2\frac{\mu_\Gamma(0)}{\gamma} \right\}.$$

We are now ready to present our main tool in proving lower bounds on the width of resolution refutations:

Theorem 3.17
For F an unsatisfiable CNF formula it holds that

$$W(F \vdash \bot) \geq \max e_\gamma(\Gamma),$$

where the maximum is taken over all sets Γ of sensitive functions compatible with F and all $\gamma \in \big(2, \mu_\Gamma(0)\big)$ for each Γ.

Proof: Given an unsatisfiable CNF formula F, fix some set Γ of sensitive functions compatible with F and a $\gamma \in \bigl(2, \mu_\Gamma(0)\bigr)$.

Suppose that π is a refutation of F. By lemma 3.10, there must exist a clause $D \in \pi$ with $\mu_\Gamma(0)/\gamma \leq \mu_\Gamma(D) \leq 2\mu_\Gamma(0)/\gamma$. Let $\Gamma' \subseteq \Gamma$ be a minimal set such that $\Gamma' \models D$. We claim that for all $x \in \partial\Gamma'$ it is also the case that $x \in \mathit{Vars}(D)$. Given this claim, we get the inequality $W(\pi) \geq W(D) \geq e_\gamma(\Gamma)$ for arbitrary resolution refutations π of F and compatible sensitive sets Γ. The theorem follows.

To prove the claim, note that for all $f \in \Gamma'$ there exists an assignment α_f such that $\alpha_f(D) = \alpha_f(f) = 0$ but $\alpha_f(g) = 1$ for $g \in \Gamma'$, $g \neq f$. (Otherwise it would hold that $\Gamma' \setminus \{f\} \models D$, but Γ' is minimal.) If $x \in \partial\Gamma' \cap \mathit{Vars}(f)$ but $x \notin \mathit{Vars}(D)$, then flipping α_f on x yields an assignment β which satisfies Γ' (by lemma 3.15) but falsifies D (since β agrees with α_f on all $y \in \mathit{Vars}(D)$). Contradiction. Consequently, $x \in \mathit{Vars}(D)$ and the claim is proved. □

3.1.3 Minimum Width Proof Search

One of the most extensively used methods for proving unsatisfiability of CNF formulas is DLL (see example 2.5 on page 24). If a CNF formula F is unsatisfiable, a DLL algorithm produces a tree refutation of F.

The results in section 3.1.1 about the length-width trade-off suggest a different method for refuting unsatisfiable formulas (which was also suggested in [4] based on an algorithm in [18]). The idea is to search for a refutation in *minimal width*. We give a rough pseudocode for the algorithm in figure 3.1 on the following page. This algorithm has a running time bounded by $n^{\mathrm{O}(W(F \vdash \perp))}$, (where as before n is the number of variables) since this is the maximal number of different clauses that will be encountered.

Thus, if a DLL algorithm produces a (tree-like) refutation of a constant-width formula F which is polynomial in the size $S(F)$, then by theorem 3.3 the running time of the algorithm above can be no worse than $S(F)^{\mathrm{O}(\log S(F))}$, i.e. it is at most quasi-polynomial in the size of F.

On the other hand, there are families of formulas which have exponential tree refutations but constant-width general resolution refutations. We give examples of this in the next section. For such formulas the above algorithm will exponentially outperform any DLL procedure.

3.2 Separations of Variants of Resolution

When presenting the resolution proof system in section 2.6, we also shortly discussed two subsystems of resolution, namely tree-like resolution and DP-resolution. There are a number of other resolution variants as well, which have been studied and for which various separations have been proved. See for instance [11] for a description of a number of resolution-based proof systems and some recent separation results.

In this thesis, we restrict our attention to general resolution $\mathcal{R}$ and tree-like resolution $\mathcal{T}$. Below, we sketch the proof of an exponential separation of $\mathcal{R}$ from $\mathcal{T}$ due to [6, 7]. We will use this result when proving separations between the dilemma and resolution proof systems in chapter 6 (theorem 6.14 on page 104).

```
w := 0
S := {C ∈ F}
while 0 ∉ S
    w := w + 1
    Derive all resolvents of width ≤ w and add to S
```

Figure 3.1: Pseudocode for minimum width proof search algorithm.

Theorem 3.18 (Exponential separation of $\mathcal{R}$ from $\mathcal{T}$)
There exists an infinite family of polynomial-size contradictory CNF formulas F_n such that $L_{\mathcal{R}}(F_n \vdash \bot) = \mathrm{O}(n)$ but $L_{\mathcal{T}}(F_n \vdash \bot) = \exp(\Omega(n/\log n))$.

The separation in theorem 3.18 is based on so called *pebbling measures* of circuits. Intuitively, the pebbling measure of a circuit is the space needed for simulating the computation of the circuit on a Turing machine.

Definition 3.19 (Circuit) *A* circuit *is a directed acyclic graph (DAG) in which each vertex has fan-in 0 or 2. A vertex with fan-in 0 is called a* source *and vertex with fan-in 2 is called a* target.

Definition 3.20 (Pebbling game) *Let G be a circuit with sources S and targets T. The* pebbling game *on G is the 1-player game described below.*

At any point in the game, some vertices of G will have pebbles on them (one pebble per vertex). A configuration *is the subset of $V(G)$ comprising just those vertices that have pebbles on them.*

The rules of the pebbling game are as follows:

- *At any time, a pebble may be placed on any vertex in S.*
- *If all immediate predecessors of a vertex v have pebbles on them, a pebble may be placed on v.*
- *At any time, a pebble may removed from any vertex.*
- *The game ends when a pebble is placed on a vertex in T.*

A legal pebbling *of T from S in G is a sequence of configurations where the first configuration is the empty set, the last one contains a vertex in T and each configuration follows from the previous one by the rules of the pebbling game. The* cost *of a legal pebbling is the maximum number of pebbles in any configuration of the pebbling. The* pebbling measure *(or* pebbling price*) of G, denoted P_G, is the minimal cost of any legal pebbling of T from S in G.*

The pebbling game is used to prove lower bounds on tree-like resolution refutations of a family of formulas known as *pebbling contradictions*.

Definition 3.21 (Pebbling contradiction) *Let G be a circuit with sources S and targets T. Associate a pair of variables $x(v)_0, x(v)_1$ with every vertex $v \in V(G)$. The* pebbling contradiction *on G, denoted Peb_G, is the conjunction of the following clauses:*

- $x(s)_0 \vee x(s)_1$ *for all* $s \in S$ *(*source axioms*),*
- *The unit clauses* $\overline{x(t)}_0$ *and* $\overline{x(t)}_1$ *for all* $t \in T$ *(*target axioms*),*
- *The four clauses* $\overline{x(u_1)}_a \vee \overline{x(u_2)}_b \vee x(v)_0 \vee x(v)_1$ *for each* $v \in V(G) \setminus S$*, where* u_1, u_2 *are the two predecessors of* v *and* a *and* b *range over* $\{0,1\}$ *(*pebbling axioms*).*

The formula Peb_G is an unsatisfiable CNF formula in width $W(Peb_G) = 4$ with $\mathrm{O}\left(\left|V(G)\right|\right)$ clauses over $2\left|V(G)\right|$ variables. It is easy to see that Peb_G has a short constant width resolution refutation.

Lemma 3.22
For a circuit G*,* $L_{\mathcal{R}}(Peb_G \vdash \perp) = \mathrm{O}\left(\left|V(G)\right|\right)$ *and* $W(Peb_G \vdash \perp) = \mathrm{O}(1)$*.*

Proof: Fix a topological sort of G and derive $x(v)_0 \vee x(v)_1$ for each $v \in V(G)$ in the order of the sort. Then resolve with the target axioms and derive the empty clause 0. It is a routine matter to verify that this refutation can be performed in width 5. □

Since the refutation in the proof above has fix width, we cannot use corollary 3.4 to get a lower bound for tree-like resolution. Instead, we relate the length of tree refutations of Peb_G to the pebbling price P_G by the next lemma (for a proof, see [6]).

Lemma 3.23
For G *a circuit,* $L_{\mathcal{T}}(Peb_G \vdash \perp) = 2^{\Omega(P_G)}$*.*

By lemma 3.23, what we need to get a separation is to find small circuits with high pebbling price. Such circuits are provided by the next lemma.

Lemma 3.24 (Paul et al. [41])
There is an infinite family of explicitly constructible circuits G_n *with* $\left|V(G)\right| = n$ *and* $P_G = \Omega(n/\log n)$*.*

Combining lemmas 3.23 and 3.24, theorem 3.18 follows.

3.3 Refutations of Random k-CNF Formulas

Random k-CNF formulas have been widely studied for a number of different reasons. Among other things, they have been used as benchmarks for satisfiability algorithms and for proving lower bounds in propositional proof systems.

It has been known for quite some time that with high probability, random k-CNF formulas from suitably chosen distributions have minimal resolution refutations of exponentially growing length [16], while other distributions yield formulas with polynomial-length refutations [27]. Below, we refer more recent results on lower and upper bounds on the length of refutations of random k-CNF formulas as a function of the *density* of the formula [3, 7].

We start by making precise the model of random k-CNF formulas which we will study.

Definition 3.25 (Random k-CNF formulas) *By a* random k-CNF formula *we mean a k-CNF formula F on n variables and $m = \Delta n$ clauses chosen at random by picking Δn clauses independently and identically distributed from the set of all $2^k\binom{n}{k}$ ordinary k-clauses with repetitions. We let $F \sim \mathcal{F}_k^{n,\Delta}$ denote that F is a formula from such a random distribution $\mathcal{F}_k^{n,\Delta}$.*

A fundamental conjecture about random k-CNF formulas (see for example [15, 16, 26, 35, 36, 39]) is the following.

Conjecture 3.26 (Existence of satisfiability threshold)
If $F \sim \mathcal{F}_k^{n,\Delta}$, there is a constant θ_k, the satisfiability threshold, *depending only on the clause width k and not on the number of variables n, such that F is satisfiable with probability $1 - \mathrm{o}(1)$ in n if $\Delta < \theta_k$ and unsatisfiable with probability $1 - \mathrm{o}(1)$ in n if $\Delta > \theta_k$.*

It is obvious that there is a lower threshold θ_k^l such that F is satisfiable with high probability if $\Delta \leq \theta_k^l$ (indeed, pick $\theta_k^l = 1/k$ to get a limit below which F is trivially guaranteed to be satisfiable). A number of authors (among others [16]) have observed that F is unsatisfiable with high probability for densities larger than the upper threshold $\theta_k^u = 2^k \ln 2$. To see this, note that a random truth assignment α satisfies F with probability $\left(1 - 2^{-k}\right)^{\Delta n}$. Since there are precisely 2^n choices for α, the probability that F is satisfiable is at most $\left(2\left(1 - 2^{-k}\right)^{\Delta}\right)^n$, which approaches zero as n goes to infinity if $\Delta > 2^k \ln 2$. Conjecture 3.26 can be reformulated as the statement that for all k, the greatest lower threshold is equal to the least upper threshold, i.e. $\theta_k^l = \theta_k^u = \theta_k$.

Such a threshold value θ_2 is known to exist for 2-CNF formulas, where $\theta_2^l = \theta_2^u = 1$ [15, 30]. For 3-CNF formulas, we only have the weaker result that $\theta_3^l \geq 3.003$ and $\theta_3^u \leq 4.571$ [26, 34, 36, 37]. Empirically, however, it seems to be the case that there is a threshold $\theta_3 \approx 4.2$ (see for example [21, 35, 38, 39]). In fact, for $k \geq 3$, [25] has shown that there is a threshold $\theta_k(n)$ for the density $\Delta = m/n$ where formulas change from being satisfiable with high probability to being unsatisfiable with high probability, but this threshold has not been proven to be independent of n.

A related question is how the difficulty of refuting an unsatisfiable random k-CNF formula F (i.e. the minimum length of a refutation) depends on the density of F. It has been shown that for fix $\Delta > \theta_k(n)$, with high probability $L_{\mathcal{R}}(F \vdash \bot) = 2^{\Omega(k_\Delta \cdot n)}$ (where $k_\Delta > 0$) [16], and if $\Delta = \Omega\left(n^{k-2}\right)$ then it holds with high probability that $L_{\mathcal{R}}(F \vdash \bot) = n^{\mathrm{O}(1)}$ [27]. But how does $L_{\mathcal{R}}(F \vdash \bot)$ change as a function of Δ and n between these limits? This is the question that will be studied below.

3.3.1 Upper Bounds on Length of Refutations

A trivial upper bound for the length of a minimum resolution refutation of a CNF formula F on n variables is $2^n + 1$ (just make a tree-like resolution derivation branching over all variables). By using fixed-order DLL, we can improve this bound to

$$L(F \vdash \bot) = 2^{\mathrm{O}\left(n/\Delta^{1/(k-2)}\right)} n^{\mathrm{O}(1)} \tag{3.1}$$

with high probability for tree-like resolution refutations (and thus general resolution refutations as well) if $F \sim \mathcal{F}_k^{n,\Delta}$. This is the next theorem.

Theorem 3.27 (Beame et al. [3])
Let $k \geq 3$ and $m = \Delta n$ where $\Delta > \theta_k(n)$ and suppose that $F \sim \mathcal{F}_k^{n,\Delta}$.

Then with probability $1 - \mathrm{o}(1)$ in n, the length of the refutation of F produced by fixed-order DLL is $2^{\mathrm{O}\left(n/\Delta^{1/(k-2)}\right)} n^{\mathrm{O}(1)}$. In particular, when $k = 3$ the refutation has length $2^{\mathrm{O}(n/\Delta)} n^{\mathrm{O}(1)}$ with probability $1 - \mathrm{o}(1)$ in n.

The proof rests on the fact that if a DLL procedure has set a sufficient number of variables, then it is quite probable that a contradiction will follow simply by unit propagation. Using this fact, we can give an upper bound for the size of the DLL tree, and hence for the length of a tree refutation, which holds with high probability.

Definition 3.28 (Critical variable) *A variable x is* critical *at a point in the execution of a DLL procedure if setting x to 0 or 1 and then applying unit propagation in either case creates the empty clause.*

Thus, if the splitting rule chooses a critical variable x the current branch of the DLL tree will terminate simply by unit propagation.

A point in the execution of a DLL procedure corresponds in a natural way to the restriction ρ defined by all variables set by the DLL procedure (including variables set by unit propagation). In the following lemma we prove that for suitably large restrictions, with high probability over half of the remaining unfixed variables are critical. This means that at the corresponding point in fixed-order DLL, it is very likely that the current branch will soon terminate.

Lemma 3.29
For any $k \geq 3$, there exists a constant c, depending only on k and not on n, such that if $F \sim \mathcal{F}_k^{n,\Delta}$, $m = \Delta n$ and ρ is a fixed restriction of t variables with $n/2 \geq t \geq c\lceil n(n/m)^{1/(k-2)}\rceil$, then with probability at least $1 - 2^{-n}$ at least half of the $n' = n - t$ unrestricted variables are critical.

Proof [of lemma 3.29]: For a k-CNF formula F and a restriction ρ, let $\widehat{C}_2(F,\rho)$ denote the set of induced 2-clauses in $F|_\rho$ on the remaining n' variables. We now give a sufficient condition for a variable $x \in \mathit{Vars}(F) \setminus \mathit{Vars}(\rho)$ to be critical in terms of the set of clauses $\widehat{C}_2(F,\rho)$.

Define the standard directed graph $G(F,\rho)$ on $2n'$ vertices, one for each literal in $\{x, \overline{x} \mid x \in \mathit{Vars}(F) \setminus \mathit{Vars}(\rho)\}$, that has directed edges $(\overline{a}, b)$ and $(\overline{b}, a)$ corresponding to each 2-clause $a \vee b$ in $\widehat{C}_2(F,\rho)$. It is clear that a sufficient condition for the variable x to be critical is that there be directed paths from x to $\overline{x}$ and from $\overline{x}$ to x, i.e. that x and $\overline{x}$ lie in the same strongly connected component of $G(F,\rho)$.

Thus, it is sufficient to show that with probability at least $1-2^{-n}$, $G(F,\rho)$ has a strongly connected component of size at least $3n'/2$. Let $C_1, C_2, \ldots, C_d$ be the strongly connected components ordered so that all edges between components go from lower to higher numbered components, and consider the first j such that $|C_1 \cup \ldots \cup C_j| \geq n'/4$. We will show that that the probability that $|C_j| < 3n'/2$ is at most 2^{-n}.

If $|C_j| < 3n'/2$ then the set $S = C_1 \cup \ldots \cup C_j$ satisfies $n'/4 \leq |S| \leq 7n'/4$ and there is no edge from the complement $\overline{S}$ to S. Consequently, to upper bound the probability that $|C_j| < 3n'/2$ it is sufficient to upper bound the probability that there is a set $S \subseteq V\big(G(F,\rho)\big)$ with $n'/4 \leq |S| \leq 7n'/4$ which is *bad* in the sense that there is no edge from $\overline{S}$ to S.

For this purpose fix a set S of size s where $n'/4 \leq s \leq 7n'/4$. The total number of distinct (ordinary) k-clauses over n variables is $2^k\binom{n}{k}$. To construct a clause C that when restricted by ρ gives an edge from $\overline{S}$ to S, we first choose $k-2$ literals among the t literals set to 0 by the restriction ρ in $\binom{t}{k-2}$ ways. Then we choose a literal a in S in s ways. Finally, to get the edge $(\overline{b}, a)$ we choose a literal $b \neq a$ such that $\overline{b} \in \overline{S}$, which can be done in at least $2n' - s - 1$ ways. It follows that the probability that a random k-clause C restricted by ρ yields an edge from $\overline{S}$ to S is at least

$$\begin{aligned}\frac{s\,(2n'-s-1)\binom{t}{k}}{2^k\binom{n}{k}} &\geq K_1\frac{(n-t)^2 t!\,k!\,(n-k)!}{(k-2)!\,(t-k+2)!\,n!} \\ &\geq K_2\frac{t(t-1)\cdots(t-k+3)}{n(n-1)\cdots(n-k+3)} \geq K_3\left(\frac{t}{n}\right)^{k-2} \\ &\geq K_4\left(\frac{cn(n/m)^{1/(k-2)}}{n}\right)^{k-2} \geq K_4 c^{k-2}\left(\frac{n}{m}\right) \qquad (3.2)\end{aligned}$$

where the (positive) constants K_i depend only on k and c is the constant mentioned in the lemma. Hence, the probability that none of the m clauses of F give an edge from $\overline{S}$ to S is at most

$$\left(1 - K_4 c^{k-2}\frac{n}{m}\right)^m \leq \exp\left(-K_4 c^{k-2} n\right) \leq 2^{-3n} \qquad (3.3)$$

if c is chosen large enough (depending only on k).

There are at most $2^{2n'}$ sets S of size s with $n'/4 \leq s \leq /7n'/4$, so the probability that there is a bad set S is at most 2^{-n}, which was to be proved. □

Using this lemma we can prove the upper bound given in the theorem.

Proof [of theorem 3.27]: Since we are interested in the *asymptotic* behaviour of the length of the refutation as a function of n and $\Delta = m/n$, without loss of generality we may assume that $m \geq (4c)^{(k-2)}n$, where c is the constant of lemma 3.29.

Let $t = cn/\Delta^{1/(k-2)}$. Then by the above assumption $t \leq n/4$. Furthermore, fixed-order DLL is clearly monotone in the number of clauses (adding extra clauses can only make it easier to refute a formula). We can therefore prove the theorem under the assumption that the number of clauses m is such that $t = \Omega(\log n)$. This is no restriction, for if m is larger we will have $2^{\mathrm{O}\left(n/\Delta^{1/(k-2)}\right)}$ smaller than $n^{\mathrm{O}(1)}$ in (3.1), and the bound will hold because of monotonicity. Since by this assumption t goes to infinity with n, it follows that it is sufficient to prove that the bound holds with probability $1 - \mathrm{o}(1)$ in t.

Fix a restriction ρ of the first t variables. We claim that the probability that there is a branch of the fixed-order DLL tree consistent with ρ that is still active (not terminated) after the first $4t$ variables have been set and the unit propagations along the way have been processed is at most 2^{-2t}. Since there are 2^t choices for ρ, using the inequality $\Pr\{\bigcup E_i\} \leq \sum \Pr\{E_i\}$ we get that with probability at least $1 - 2^{-t}$, every branch of the DLL tree is completed after at most the first $4t$ variables have been set and the resulting unit propagations done. It follows that with the same asymptotic probability the tree has at most $2n2^{4t}$ nodes (including nodes from unit propagation).

It remains to prove the claim. By lemma 3.29, the probability that the size r of the set of critical variables for $F|_\rho$ is less than $n'/2$ is at most $2^{-n} \leq 2^{-4t}$. Suppose that $r \geq n'/2$. Then the set of critical variables is equally likely to be any r-subset of the $n' = n - t$ unfixed variables. The probability that none of the next $3t$ variables in order are critical is at most

$$\begin{aligned} \binom{n'-3t}{r} \Big/ \binom{n'}{r} &= \frac{(n'-3t)!\,(n'-r)!}{n'!\,(n'-3t-r)!} \\ &= \frac{(n'-3t)(n'-3t-1)\cdots(n'-3t-r+1)}{n'(n'-1)\cdots(n'-r+1)} \leq \left(\frac{n'-3t}{n'}\right)^r \\ &= \left(1-\frac{3t}{n'}\right)^r \leq \left(1-\frac{3t/2}{n'/2}\right)^{n'/2} \leq e^{-3t/2}. \end{aligned} \tag{3.4}$$

Hence the probability that some branch consistent with ρ is unfinished after fixing the next $3t$ variables is at most $2^{-4t} + e^{-3t/2} \leq 2^{-2t}$, which proves the claim. □

3.3.2 Lower Bounds on Length of Refutations

We now turn to the question of giving lower bounds on the length of refutations of random k-CNF formulas. We prove the bound given in the following theorem by using the results in section 3.1 relating lower bounds on refutation length and refutation width.

Theorem 3.30 (Beame et al. [3])
For $F \sim \mathcal{F}_3^{n,\Delta}$ and any $\epsilon > 0$, with probability $1 - o(1)$ in n it holds that

$$L_{\mathcal{R}}(F \vdash \bot) = \exp\big(\Omega\left(n/\Delta^{4+\epsilon}\right)\big).$$

To prove this lower bound, we look at the expansion properties of hypergraphs H_F which we associate with every k-CNF formula F. The vertices of H_F are the variables in F and each clause of F defines a hyperedge.

Definition 3.31 *For a k-CNF formula F with $m = \Delta n$ clauses on n variables, let H_F be the hypergraph with vertices $V(H_F)$ and edges $E(H_F)$ given by*

$$\begin{aligned} V(H_F) &:= [n], \\ E(H_F) &:= \big\{(i_1,\ldots,i_k) \mid \exists C \in F : \{x_{i_1},\ldots,x_{i_k}\} = \mathit{Vars}(C)\big\}. \end{aligned}$$

For any subset V' of vertices, let $E(V')$ denote the set of edges within V',

$$E(V') := \big\{(i_1,\ldots,i_k) \in E(H_F) \mid i_1,\ldots,i_k \in V'\big\},$$

and for any subset E' of edges, let $V(E')$ denote the set of vertices covered by E',

$$V(E') := \big\{i \in V(H_F) \mid \exists i_2,\ldots,i_k : (i,i_2,\ldots,i_k) \in E'\big\}.$$

We will need a special definition for expansion suited for hypergraphs.

Definition 3.32 (Hypergraph expansion) *Let H be a k-regular hypergraph with n vertices. The* hypergraph δ-expansion *(or just* δ-expansion*) of H is*

$$e_\delta(H) := \min \left\{ 2 \cdot \left|V(E')\right| - k \cdot |E'| : E' \subseteq E(H),\ \delta n \leq |E'| \leq 2\delta n \right\}.$$

However, positive expansion of H_F will not alone be sufficient to ensure that the width of refuting F is large. We also need to know that there is not a small unsatisfiable subformula of F. This is taken care of by the next definition.

Definition 3.33 (λ-matchability) *A k-regular hypergraph H with n vertices is* λ-matchable $(0 < \lambda \leq 1)$ *if for all subsets of edges $E' \subseteq E(H)$ such that $|E'| \leq \lambda n$ it holds that $\left|V(E')\right| \geq |E'|$.*

The main theorem of this section states that for a k-CNF formula F with a "sufficiently matchable" hypergraph H_F (i.e. with sufficiently large λ in definition 3.33 above), the width of refuting F is bounded from below by the hypergraph expansion of H_F.

Theorem 3.34
Let F be an unsatisfiable k-CNF formula with $m = \Delta n$ clauses on n variables such that the associated hypergraph H_F is $1/\Delta^2$-matchable. Then

$$W(F \vdash \bot) \geq e_\delta(H_F)$$

for any $\delta \leq 1/(2\Delta^2)$.

Proof: Set $\Gamma := \{C \in F\}$. Certainly, the set of clauses of F is a set of sensitive functions (definition 3.12) compatible with F (definition 3.9).

Since H_F is $1/\Delta^2$-matchable, $\mu_\Gamma(0) > n/\Delta^2$. To see this, note that by the $1/\Delta^2$-matchability of H_F, for any $\Gamma' \subseteq \Gamma$ with $|\Gamma'| \leq n/\Delta^2$ it holds that $|\mathit{Vars}(\Gamma')| \geq |\Gamma'|$. By Hall's marriage theorem, this means that for any Γ' with $|\Gamma'| \leq n/\Delta^2$ there is a perfect matching between the clauses in Γ' and the variables in $\mathit{Vars}(\Gamma')$. Using such a matching, we can satisfy all clauses in Γ' by setting the variables to appropriate values so that every variable satisfies the clause to which it has been matched. Hence $\mu_\Gamma(0) > n/\Delta^2$ as claimed.

Now let π be a refutation of F. By the compatibility of Γ and lemma 3.10, for any $\delta \leq 1/(2\Delta^2)$ there is a clause $D \in \pi$ with $\delta n \leq \mu_\Gamma(D) \leq 2\delta n$ $(\leq n/\Delta^2)$. For such a D, let $\Gamma' \subseteq \Gamma$ be a minimal set of clauses such that $\Gamma' \models D$. Every variable outside the boundary of Γ' (definition 3.14) must be covered by at least two clauses of Γ', so

$$|\mathit{Vars}(\Gamma')| \leq |\partial\Gamma'| + \tfrac{1}{2}\left(k \cdot |\Gamma'| - |\partial\Gamma'|\right) \tag{3.5}$$

or equivalently

$$|\partial\Gamma'| \geq 2 \cdot |\mathit{Vars}(\Gamma')| - k \cdot |\Gamma'|. \tag{3.6}$$

If we let E' denote the edges in H_F corresponding to the clauses in Γ' and switch to hypergraph notation, the inequality (3.6) becomes

$$|\partial\Gamma'| \geq 2 \cdot \left|V(E')\right| - k \cdot |E'|. \tag{3.7}$$

Thus, for a suitably chosen γ (namely $\gamma = \mu_\Gamma(0)/\delta n$), it holds that $e_\gamma(\Gamma) \geq e_\delta(H_F)$. By theorem 3.17, $W(F \vdash \bot) \geq e_\gamma(\Gamma) \geq e_\delta(H_F)$, which concludes the proof. $\square$

For 3-CNF formulas we have the following lemma (which can be extended to k-CNF formulas for any fixed k [2, 3]).

Lemma 3.35
For all $\epsilon > 0$ there is a constant $c_\epsilon > 0$, not dependent of n, such that if $\Delta < n^{1/2-\epsilon}$ and H is a uniformly random 3-regular hypergraph with n vertices and $m = \Delta n$ edges, then with probability $1 - \mathrm{o}(1)$ in n the hypergraph H is $1/\Delta^2$-matchable and $e_\delta(H) \geq c_\epsilon n/\Delta^{2+\epsilon}$ (for a suitably chosen δ).

Lemma 3.35 is proved by calculating a union bound using the techniques in [2, 7]. Theorem 3.34 in combination with lemma 3.35 yields:

Theorem 3.36
For $F \sim \mathcal{F}_3^{n,\Delta}$ and any $\epsilon > 0$, with probability $1 - \mathrm{o}(1)$ in n it holds that

$$W_{\mathcal{R}}(F \vdash \bot) = \exp\big(\Omega\left(n/\Delta^{2+\epsilon}\right)\big).$$

If we plug in the width bound in the last theorem in corollary 3.6 on page 30, we get the bound stated in theorem 3.30 at the beginning of this section.

Chapter 4

Stålmarck's Method

So far in this Master's thesis, we have given a general introduction to proof theory and reviewed some current results for the resolution proof system. It is now time to attend to the main subject of the thesis, namely Stålmarck's proof method and the propositional proof system on which it is based.

Stålmarck's method is defined in terms of a system for natural deduction incorporating the subformula principle. A central concept in this system is the proof-theoretic measure of *proof depth*, that is, the maximal number of nested assumptions in a proof. This measure turns out to capture the hardness of proving a tautology F in the sense that if F has a short proof, then there is also a *shallow* proof of F. In view of this, a natural strategy when trying to prove a tautology is to search for shallow proofs. Stålmarck's proof procedure pursues this strategy by implementing an algorithm called *κ-saturation*, which performs efficient exhaustive search for proofs in fixed depth κ.

The propositional proof system underlying Stålmarck's method can roughly be described as consisting of two components. Firstly, it includes all derivation rules that do not discharge assumptions but derive consequences which follow "immediately" from the truth tables for the logical connectives, so called *propagation rules*. In order to exploit fully the power of the propagation rules, derivations are defined not in terms of sequences of formulas but sequences of *formula equivalence relations*. Secondly, the system contains a discharge rule embodying the bivalence principle (i.e. that the logic is two-valued). Loosely put, the rule splits over subexpressions of the formula to be proved and investigates what consequences can be derived in the two branches assuming the subexpressions true and false, respectively. The results are then merged by concluding all consequences derived in both branches (they must be true since they hold regardless of the truth or falsehood of the assumptions in the branches).

The discharge rule is known as the *dilemma rule*. It is the design of this rule which gives proofs produced by Stålmarck's method their characteristic series-parallel shape. In this way, one can avoid duplication of subderivations at different places in the proof which can lead to exponential blow-up in tree-shaped proofs. Because of the importance of this, the proof system itself has taken its name from the dilemma rule and is consequently called the *dilemma proof system* (or just *dilemma* for short).

This chapter is divided into two parts. In section 4.1, we introduce the dilemma proof system. In section 4.2, Stålmarck's algorithm for exhaustive proof search in this system is presented. Our exposition of the dilemma proof system and Stålmarck's method relies heavily on the tutorial by Sheeran and Stålmarck [49]. The material presented here expands and formalizes the concepts in [49], however, and also introduces new, more precise terminology and notation. This was necessitated by the need to be able to reason strictly formally about the proof system when proving the results in chapter 6. To our knowledge, this thesis contains the first complete explicit formal description of the dilemma proof system.

4.1 The Proof System

This section is organized as follows. The formula equivalence relations which are the building blocks of derivations in dilemma are introduced in section 4.1.1. The propagation and dilemma rules which together constitute the derivation rules of the proof system are discussed in sections 4.1.2 and 4.1.3, respectively, after which a formal description of the dilemma proof system is given in section 4.1.4. In section 4.1.5, we investigate the relation between proof depth and proof length and introduce the measure of *proof hardness*, according to which formulas can be classified as hard or easy. We conclude our presentation of dilemma in section 4.1.6 by a short discussion of the hierarchy resulting from ordering formulas with respect to proof hardness.

4.1.1 Formula Relations

In the proof systems presented in chapter 2, we keep track on formulas that are true (and false, if we negate true formulas) and use the derivation rules to derive new true (and false, respectively) formulas.

The dilemma proof system uses a larger set of rules for which it is no longer sufficient to maintain only sets of formulas known to be true or false. We must also maintain information about pairs of formulas whose truth values are as yet unknown but which have been showed to have the same truth value or opposite truth values. To this end we introduce *formula relations*.

A formula relation is an equivalence relation over the subformulas of a logical formula F and their complements which in addition to being reflexive, symmetric and transitive also respects the semantic interpretation of logical negation. (We remind that the *complement* of a formula P as defined in definition 2.6 on page 10 is $\neg P$ if P is not a negation and P with the outermost negation removed otherwise.)

Here and in the following, we assume that our formulas F contain no double negations, i.e. that there is no $\neg P \in \mathit{Sub}(F)$ such that $\neg(\neg P) \in \mathit{Sub}(F)$. (An alternate way of seeing this is to imagine a "preprocessor step" before a formula F is passed to a proof system or proof search algorithm, in which all occurrences of $\neg(\neg P)$ are replaced by P.) While this restriction is not strictly necessary, it simplifies the proofs of some theorems somewhat. In particular, it has the consequence that $\left(P^C\right)^C = P$.

Definition 4.1 (Formula relation) *Let F be a formula in propositional logic. A* formula relation *on F is an equivalence relation $\equiv$ with domain $\mathit{Sub}(F)$,* *subject to the additional constraint that if $P \equiv Q$ holds, it must also hold that $P^C \equiv Q^C$ (where we adopt the convention that $\top^C = \bot$).*

We use the letter R to denote an arbitrary formula relation and C and D to denote equivalence classes of formula relations.

Definition 4.2 *If R is a formula relation on F, we define $Vars(\mathsf{R}) := Vars(F)$, $Sub\,(\mathsf{R}) := Sub\,(F)$ and $Compound\,(\mathsf{R}) := Compound\,(F)$.*

*We call the equivalence class containing $\top$ the TRUE-*class *and the class containing $\bot$ the FALSE-*class*. These are the* determinate *classes. The remaining classes are called* indeterminate.

If $\mathsf{C} = \{P_1, \ldots, P_n\}$ is an equivalence class of R, the complement *of C is $\mathsf{C}^C = \{{P_1}^C, \ldots, {P_n}^C\}$.*

Note that if $\mathsf{C} \in \mathsf{R}$, then we also have $\mathsf{C}^C \in \mathsf{R}$ (since formula relations respect complement).

If P and Q are in the same equivalence class of a formula relation, this means that (under the assumptions made) they must have the same truth value. By defining formula relations on $Sub\,(F)$, which includes the complements of all subformulas of F, we can encode both equality and inequality of truth values. The fact that P and Q have opposite truth values is encoded as $P \equiv Q^C$.

In order for our definition of dilemma derivations as sequences of formula relations to be meaningful, the formula relations must be defined over the same domain.

Definition 4.3 (Compatibility) *Two formula relations R_1 and R_2 are said to be* compatible *if they are defined on the same formula F, i.e. if it holds that $Sub\,(\mathsf{R}_1) = Sub\,(\mathsf{R}_2)$.*

Compatible formula relations can be compared. The more subformulas have been related in a formula relation, the stronger the relation is.

Definition 4.4 *We say that R_2 is a* stronger *or* larger *formula relation than R_1, denoted $\mathsf{R}_1 \sqsubseteq \mathsf{R}_2$, if R_1 and R_2 are compatible and every equivalence class C_1 of R_1 is a subset of an equivalence class C_2 of R_2.*

If $\mathsf{R}_1 \sqsubseteq \mathsf{R}_2$ and $\mathsf{R}_1 \neq \mathsf{R}_2$, we say that R_2 is strictly stronger (larger) *than R_1 and write $\mathsf{R}_1 \sqsubset \mathsf{R}_2$.*

In the dilemma proof system we prove (or disprove) formulas by gradually enlarging formula relations on the formula in question, or, phrasing it somewhat differently, by adding new *associations*.

Definition 4.5 (Association) *For R a formula relation and $P, Q \in Sub\,(\mathsf{R})$, we write $\mathsf{R}\big[P \equiv Q\big]$ to denote the least formula relation which contains R and* relates *or* associates *P and Q (that is, merges the equivalence classes of P and Q). We call $P \equiv Q$ an* association *on R. If ψ is the association $P \equiv Q$, then ψ^C denotes the complementary association $P \equiv Q^C$.*

We extend the notation for additions to relations from single associations to sets of associations in the obvious way. If $\Psi = \{\psi_1, \ldots, \psi_n\}$ is a set of associations, then $\mathsf{R}\big[\Psi\big]$, which we write $\mathsf{R}\big[\psi_1, \ldots, \psi_n\big]$, is $(\mathsf{R}\big[\psi_1\big])\big[\psi_2, \ldots, \psi_n\big]$. Also, if Ψ and Φ are two sets of associations, then we will let $\mathsf{R}\big[\Psi, \Phi\big]$ denote the relation $\big(\mathsf{R}\big[\Psi\big]\big)\,\big[\Phi\big]$.

It is trivial to show that formula relation associations are commutative.

The smallest formula relation on a formula F is the identity relation on $Sub(F)$, which we denote F^+. It simply places each element in $Sub(F)$ in its own equivalence class. Two rather more interesting formula relations are $F^+[F \equiv \top]$, which we abbreviate by $F^\top$, and $F^+[F \equiv \bot]$, abbreviated $F^\bot$. These two relations will be the starting points when we try to prove that F is a contradictory or valid formula, respectively.

Definition 4.6 *If F is a contradictory (or valid) formula, we say that the relation $F^\top$ (or $F^\bot$, respectively) is* (implicitly) contradictory. *A formula relation* R *is said to be* explicitly contradictory *if one of the equivalence classes in* R *contains both a subformula and its complement.*

Perhaps the simplest example of an explicitly contradictory formula relation is the relation $F^+[F \equiv F^C]$.

4.1.2 Propagation Rules

Given a formula relation R, we can inspect subformulas $P \circ Q \in Sub(\mathsf{R})$ (for connectives $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$) to see if the semantics of the connective $\circ$ can be exploited to derive new associations on R based on already known truth values or relations between truth values of P, Q and $P \circ Q$. Such derivations propagate information which is already "explicit" in R without discharging any assumptions, and the rules for making them are consequently called *propagation rules* (or *simple rules*).

Definition 4.7 (Propagation rule) *A* propagation rule *(or* simple rule*) for a binary logical connective $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ is a rule for how one of the formulas in the set $M = \{P, Q, P \circ Q\}$ can be unambiguously associated to $\top$ or $\bot$ (that is, assigned a truth value) or to another formula or complement of another formula in M, given that the truth value of one or two of the other formulas in M are known or that the relation between the truth values of two of the formulas in M has been determined.*

The propagation rules for a connective $\circ$ can be described by schemas

$$\frac{U_1 \equiv V_1, \ldots, U_n \equiv V_n}{U \equiv V} \tag{4.1}$$

where $U_i, V_i, U, V \in \{P, Q, P \circ Q, P^C, Q^C, \top, \bot\}$. Each such schematic rule can be seen to correspond to a partial function on formula relations that takes a relation R in which $U_i \equiv V_i$ for $i = 1, \ldots, n$ and returns the relation $\mathsf{R}[U \equiv V]$.

When picking a set of propagation rules for a proof system, we include only *proper* rules. A propagation rule is proper if

$$\{U_1 \equiv V_1, \ldots, U_n \equiv V_n\} \models U \equiv V \tag{4.2}$$

$$\{U_1 \equiv V_1, \ldots, U_n \equiv V_n\} \not\models \bot \tag{4.3}$$

$$\{U_1 \equiv V_1, \ldots, U_n \equiv V_n\} \setminus \{U_i \equiv V_i\} \not\models U \equiv V \tag{4.4}$$

For the dilemma proof system, a systematic study of the semantics of the logical connectives reveals that $n = 1$ suffices in the rule schema (4.1) for all propagation rules (see appendix A). We will make use of this observation later in this chapter.

In dilemma, at each propagation step in a derivation we apply to a formula relation R the partial function corresponding to some propagation rule in order to derive new information about equivalences between the subformulas in R. The principle behind the set of propagation rules in the dilemma proof system is that whenever subformulas $P \circ Q$, P and Q are related to each other, to complements of each other or to $\top$ and $\bot$ in such a way that the semantics of the connective $\circ$ imply that further equivalences between the subformulas can be derived, there is a rule on the form (4.1) to derive that equivalence.

We now give some examples of propagation rules, deferring a discussion of the rather extensive full set of propagation rules to appendix A.

Two examples of (symmetric pairs of) propagation rules for conjunction are the elimination rules

$$\frac{P \wedge Q \equiv \top}{P \equiv \top} \qquad \frac{P \wedge Q \equiv \top}{Q \equiv \top} \tag{4.5}$$

and the "negative" introduction rules

$$\frac{P \equiv \bot}{P \wedge Q \equiv \bot} \qquad \frac{Q \equiv \bot}{P \wedge Q \equiv \bot}\,. \tag{4.6}$$

The corresponding rules for disjunction are the elimination rules

$$\frac{P \vee Q \equiv \bot}{P \equiv \bot} \qquad \frac{P \vee Q \equiv \bot}{Q \equiv \bot} \tag{4.7}$$

and the introduction rules

$$\frac{P \equiv \top}{P \vee Q \equiv \top} \qquad \frac{Q \equiv \top}{P \vee Q \equiv \top}\,. \tag{4.8}$$

Note that the above rules (or variants of them) can be found also in the proof systems presented in chapter 2. We see that using only information about true or false subformulas to derive new true of false formulas corresponds to restricting V_i and V to the set $\{\top, \bot\}$ in the rule schema (4.1).

As mentioned above, the dilemma proof system extends the set of propagation rules by allowing V_i and V to be arbitrary subformulas of the formula to be refuted or proved valid.

Examples of such rules for conjunction based on information about subformula equivalences are the rules

$$\frac{P \wedge Q \equiv P^C}{P \equiv \top} \qquad \frac{P \wedge Q \equiv Q^C}{Q \equiv \top} \tag{4.9}$$

$$\frac{P \wedge Q \equiv P^C}{Q \equiv \bot} \qquad \frac{P \wedge Q \equiv Q^C}{P \equiv \bot} \tag{4.10}$$

(if a conjunction of two subformulas is equivalent to the complement of one of the subformulas we know the truth values of both subformulas),

$$\frac{P \equiv Q}{P \wedge Q \equiv P} \qquad \frac{P \equiv Q}{P \wedge Q \equiv Q} \tag{4.11}$$

(if two subformulas are equivalent, then the conjunction of them is equivalent to either subformula) and

$$\frac{P \equiv Q^C}{P \wedge Q \equiv \bot} \tag{4.12}$$

(if two subformulas have different truth values the conjunction of them is certainly false).

For disjunction the corresponding rules are:

$$\frac{P \vee Q \equiv P^C}{P \equiv \bot} \qquad \frac{P \vee Q \equiv Q^C}{Q \equiv \bot} \tag{4.13}$$

$$\frac{P \vee Q \equiv P^C}{Q \equiv \top} \qquad \frac{P \vee Q \equiv Q^C}{P \equiv \top} \tag{4.14}$$

$$\frac{P \equiv Q}{P \vee Q \equiv P} \qquad \frac{P \equiv Q}{P \vee Q \equiv Q} \tag{4.15}$$

$$\frac{P \equiv Q^C}{P \vee Q \equiv \top} \tag{4.16}$$

A complete set of propagation rules in the dilemma proof system for the logical connectives $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$ is given in section A.2 of appendix A. We refer to

- figure A.2 on page 127 for conjunction $\wedge$,
- figure A.3 on page 127 for disjunction $\vee$,
- figure A.4 on page 128 for implication $\rightarrow$,
- figure A.5 on page 129 for bi-implication $\leftrightarrow$.

No special rules for negation are needed because of the constraint that formula equivalences should respect complement. One way of looking at this constraint is as an implicit application of the *complement rule*

$$\frac{P \equiv Q}{P^C \equiv Q^C} \tag{4.17}$$

after each step in a derivation.

In dilemma, reaching a contradiction means deriving an explicitly contradictory formula relation, i.e. a relation on the form $\mathsf{R}[\psi, \psi^C]$. To simplify the reasoning, we will usually assume that an explicitly contradictory relation on a formula F is always extended (in one derivation step) to the relation on F consisting of only one equivalence class by the rule

$$\frac{P \equiv P^C}{Q \equiv \top} \quad \forall Q \in \mathit{Sub}\,(F) \tag{4.18}$$

which we will call the *contradiction rule*.

For convenience, we will sometimes treat the complement and contradiction rules as simple rules. It should be observed, however, that these rules are *not* simple rules in the strict sense of definition 4.7.

4.1.3 The Dilemma Rule

Formula relations together with the large set of propagation rules for them that we have introduced allow us to reach conclusions which would require branching in proof systems with a smaller set of propagation rules. In this way we can minimize the amount of branching, something which is of great importance. As we shall see when we prove bounds on proof length (section 4.1.5) and analyze Stålmarck's method (section 4.2.2) later in this chapter, it is branching that is the critical factor for proof complexity and efficiency of the proof search algorithm.

Of course, branching cannot be avoided altogether, since the simple rules discussed in section 4.1.2 are not complete for propositional logic. The branching rule used in the dilemma proof system is a special rule for branching *and merging* called the *dilemma rule*. Schematically it can be described as:

$$\frac{\mathsf{R}}{\dfrac{\begin{array}{cc} \mathsf{R}\big[P \equiv Q\big] & \mathsf{R}\big[P \equiv Q^C\big] \\ \pi_1 & \pi_2 \\ \mathsf{R}_1 & \mathsf{R}_2 \end{array}}{\mathsf{R}_1 \sqcap \mathsf{R}_2}} \tag{4.19}$$

This rule schema is to be interpreted as follows: Given a formula relation R on a formula F, we apply the dilemma rule by choosing subformulas P and Q of F from different (and non-complementary) equivalence classes in R. Next, we make two new dilemma derivations starting from the relations $\mathsf{R}\big[P \equiv Q\big]$ and $\mathsf{R}\big[P \equiv Q^C\big]$, deriving R_1 and R_2, Finally, we conclude the formula relation intersection $\mathsf{R}_1 \sqcap \mathsf{R}_2$ of the two formula relations, that is the formula relation containing all common associations in R_1 and R_2 (a formal definition is given below). This corresponds to the fact that equivalences derived both under the assumption $P \equiv Q$ and under the assumption $P \equiv Q^C$ must hold regardless of the truth values of P and Q. We call the equivalences $P \equiv Q$ and $P \equiv Q^C$ above *dilemma rule assumptions* and instances of rule (4.19) *dilemma rule applications*.

Definition 4.8 (Formula relation intersection) *Let* R_1 *and* R_2 *be compatible formula relations. The* formula relation intersection $\mathsf{R}_1 \sqcap \mathsf{R}_2$ *of* R_1 *and* R_2 *is the largest formula relation* R *for which* $\mathsf{R} \sqsubseteq \mathsf{R}_1$ *and* $\mathsf{R} \sqsubseteq \mathsf{R}_2$.

Alternatively, we may define $\mathsf{R}_1 \sqcap \mathsf{R}_2$ *by*

$$\mathsf{R}_1 \sqcap \mathsf{R}_2 := \big\{\{\mathsf{C}_1 \cap \mathsf{C}_2\} \mid \big(\mathsf{C}_1, \mathsf{C}_2\big) \in \mathsf{R}_1 \times \mathsf{R}_2\big\} \setminus \{\emptyset\}.$$

If we reach a contradiction in, say, the left branch of a dilemma rule application, by the contradiction rule (4.18) we have $\mathsf{R}_1 \sqcap \mathsf{R}_2 = \mathsf{R}_2$, and closing the application of the dilemma rule corresponds to concluding all equivalences derived in the right branch. Thus, this special case of the dilemma rule is just a slightly more complicated way of expressing the derivation rule *reductio ad absurdum* or *RAA*.

Thanks to the more general formulation of rule (4.19), however, we can use results deduced in both branches even when no contradiction has been derived. This makes our proof system much more efficient. If we would omit the merging of the branches and refute R_1 and R_2 separately, then the two subproofs would probably to have much in common (since R_1 and R_2 have R in common). Merging the two branches avoids this repetition.

4.1.4 Definition of Dilemma Derivation

We now define formally[1] what we mean by a derivation in the dilemma proof system, or a *dilemma derivation*. We also define the proof-theoretic measures of depth and length of dilemma derivations and discuss the rather close relationship between length and size in dilemma (in contrast to for example in resolution).

Definition 4.9 (Dilemma derivation) *Let F be a formula in propositional logic. Let R, R_i denote formula relations on F and suppose that ψ is an association on R.*

Simple derivation *If an application of one of the simple rules in figures A.2, A.3, A.4 or A.5 (followed by automatic closure under complement by rule* (4.17)*) or of the contradiction rule* (4.18) *to the relation R_1 yields the relation R_2 (where $\mathsf{R}_1 \sqsubset \mathsf{R}_2$) then*

$$\pi = \begin{array}{c} \mathsf{R}_1 \\ \mathsf{R}_2 \end{array}$$

is a (simple) dilemma derivation *of* R_2 *from* R_1 *(or a* dilemma derivation step *from* R_1 *to* R_2*). As a special case,*

$$\pi' = \mathsf{R}$$

is a derivation of R from R. Here and in the following, we write the assertion that π is a dilemma derivation of R_2 from R_1 as $\pi : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$.

The derivation depth *$D(\pi)$ of a simple derivation π is zero. The* derivation length *of the simple derivation $\pi : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ above is $L(\pi) = 2$ and the length of $\pi' : \mathsf{R} \Rightarrow \mathsf{R}$ is $L(\pi') = 1$.*

Composition *If $\pi_1 : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ and $\pi_2 : \mathsf{R}_2 \Rightarrow \mathsf{R}_3$ (where $\mathsf{R}_1 \sqsubset \mathsf{R}_3$) then the* composition *$\pi_1 \bullet \pi_2$ of π_1 and π_2 defined by*

$$\pi_1 \bullet \pi_2 = \begin{array}{c} \pi_1 \\ \pi_2 \end{array}$$

(containing only one copy of the intermediate relation R_2) is a dilemma derivation $\pi_1 \bullet \pi_2 : \mathsf{R}_1 \Rightarrow \mathsf{R}_3$.

The depth of a composition $\pi_1 \bullet \pi_2$ is $D(\pi_1 \bullet \pi_2) = \max\{D(\pi_1), D(\pi_2)\}$ and the length is $L(\pi_1 \bullet \pi_2) = L(\pi_1) + L(\pi_2) - 1$.

Dilemma rule *If $\pi_1 : \mathsf{R}[\psi] \Rightarrow \mathsf{R}_1$ and $\pi_2 : \mathsf{R}[\psi^C] \Rightarrow \mathsf{R}_2$ then*

$$\pi = \frac{\mathsf{R}}{\dfrac{\pi_1 \qquad \pi_2}{\mathsf{R}_1 \sqcap \mathsf{R}_2}}$$

[1] As remarked in chapter 2, in order to be formally correct a proof in a propositional proof system should include annotations for each step by which rules it was derived so that the validity of the proof can be efficiently checked. However, such annotations would change the proof size only by a small constant factor. Therefore they are of little theoretical or practical interest and we have omitted them in the definition of the dilemma proof system.

(where $\mathsf{R}[\psi] \neq \mathsf{R} \neq \mathsf{R}[\psi^C]$ *and* $\mathsf{R} \sqsubset \mathsf{R}_1 \sqcap \mathsf{R}_2$*) is a* dilemma rule derivation *(or* dilemma rule application*)* $\pi : \mathsf{R} \Rightarrow \mathsf{R}_1 \sqcap \mathsf{R}_2$.

The depth of $\pi : \mathsf{R} \Rightarrow \mathsf{R}_1 \sqcap \mathsf{R}_2$ *is* $D(\pi) = \max\{D(\pi_1), D(\pi_2)\} + 1$ *and the length is* $L(\pi) = L(\pi_1) + L(\pi_2) + 2$.

We see that the depth of a dilemma derivation is the maximum number of simultaneously open branches, and the length of a derivation is the number of occurrences (with repetitions) of formula relations in it.

As for proof systems in general, the *size* $S(\pi)$ of a dilemma derivation π is defined to be the total number of symbols in the derivation. In the dilemma proof system a full formula relation on F is explicitly given at each step in a derivation based on the formula F. If we encode formula relations in the naive way[2] by writing out each subformula in the relation in full, the number of symbols needed is $\Omega(S(F))$ and $\mathrm{O}(S(F)^2)$, so the size $S(\pi)$ of a dilemma derivation π based on the formula F is related to the length $L(\pi)$ by

$$L(\pi) \cdot \Omega(S(F)) \leq S(\pi) \leq L(\pi) \cdot \mathrm{O}\left(S(F)^2\right). \tag{4.20}$$

The interpretation of this inequality is that for all practical purposes, the measures of length and size of dilemma derivations are interchangeable. (Note that this is not true for proof systems in general. In particular, it does not hold for resolution.)

Our definition of dilemma derivations explicitly disallows redundant rule applications (a redundant rule application is one that does not draw any new conclusions, so that the concluding relation is the same as the starting one). In some contexts it can be convenient to relax the demand on inequality between the formula relations in the definition above and allow such redundancy. Also, in some contexts it is advantageous to ignore the convention that the derivation of an explicitly contradictory formula relation should always be followed by an application of the contradiction rule (4.18). In order to be able to make these distinctions, we introduce some new terminology.

Definition 4.10 *A derivation formed according to the rules in definition 4.9 except for the fact that* $\mathsf{R}_1 = \mathsf{R}_2$ *for some simple derivation,* $\mathsf{R}_1 = \mathsf{R}_3$ *for some composition or* $\mathsf{R} = \mathsf{R}_1 \sqcap \mathsf{R}_2$ *for some dilemma rule application is called a* non-proper *dilemma derivation. Derivations conforming strictly to definition 4.9 are called* proper *derivations.*

A dilemma derivation (proper or non-proper) where the derivation of an explicitly contradictory formula relation is always immediately followed by an application of the contradiction rule (4.18) *is called a* standardized *derivation. A derivation for which this is not the case is called* non-standardized.

We say that a dilemma derivation (proper or non-proper) $\pi : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ *is* nontrivial *if* $\mathsf{R}_1 \neq \mathsf{R}_2$ *and* trivial *otherwise.*

The proof of the following technical proposition is easy.

[2]In section 4.2.1 we introduce a more efficient representation of formula relations in size $\mathrm{O}(S(F))$. However, since a factor $S(F)$ in size does not affect our results in any significant way, we choose the more straightforward $\mathrm{O}(S(F)^2)$-representation for the theoretical analysis.

Proposition 4.11

1. *Any non-proper dilemma derivation $\pi' : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ can be transformed to an equivalent proper derivation $\pi : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ of at most the same depth, length and size.*

2. *Any non-standardized dilemma derivation $\pi' : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ (which can be proper or non-proper) can be transformed to an equivalent proper, standardized derivation $\pi : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ with $D(\pi) \leq D(\pi')$, $L(\pi) \leq \frac{3}{2} L(\pi')$ and $S(\pi) \leq \frac{3}{2} S(\pi')$.*

In view of this proposition, it is often sufficient in proofs of theorems about the dilemma proof system to reason in terms of non-proper, non-standardized dilemma derivations since such derivations can easily be transformed to proper, standardized derivations of essentially the same size and shape. Usually, the constant factor involved in the transformation is of no consequence.

We have defined derivations in the dilemma proof system in terms of *relations*, not formulas. To prove something in the dilemma proof system, we start from a formula relation corresponding to the negation of what is to be proved and derive a contradiction. Thus, a *dilemma proof* is a dilemma derivation which ends by deriving an explicitly contradictory formula relation.

To make the connection back from derivations involving formula relations to proofs about formulas, note that the derivation of an explicitly contradictory formula relation from $F^\top$ constitutes a refutation of the formula F. Similarly, we can prove that F is a tautology by refuting the formula relation $F^\perp$. At times, the distinction between refutations and (tautology) proofs of formulas tends to become rather blurred. This is quite natural, since proving F valid corresponds to refuting $\neg F$. However, in this thesis we will try to stick to the terminology presented in the following definition.

Definition 4.12 (Dilemma proof and refutation) *Suppose that R is a formula relation and let $\perp_\mathsf{R}$ denote the canonical explicitly contradictory formula relation on the domain of R consisting of only one equivalence class.*

A dilemma proof *from the formula relation R (or* dilemma refutation *of the relation R) is a dilemma derivation $\pi : \mathsf{R} \Rightarrow \perp_\mathsf{R}$.*

A dilemma proof *of the formula F (a proof that F is valid) is a dilemma derivation $\pi : F^\perp \Rightarrow \perp_{F^+}$.*

A dilemma refutation *of the formula F (a proof that F is contradictory) is a dilemma derivation $\pi : F^\top \Rightarrow \perp_{F^+}$.*

Remark 4.13 In dilemma as defined by definitions 4.9 and 4.12, one cannot derive anything without premises. Thus, the dilemma proof system is not complete and does not in itself constitute a natural deduction system in the strict sense of the word. However, it is *refutation complete* for formulas in propositional logic. That is, given an unsatisfiable (or tautological) formula F, we can refute (or prove) F in dilemma by making a derivation of $\perp_{F^+}$ from $F^\top$ (or $F^\perp$, respectively).

We could easily remedy this by changing definition 4.12 so that a dilemma derivation π starts from a formula relation F^+ with no premises and constitutes a proof of all subformulas of F found in the *TRUE*-class at the end of the derivation (and, if we wish, change definition 4.9 so that we start with an empty formula relation and let its domain grow by adding $\mathit{Sub}(P) \cup \mathit{Sub}(Q)$ for each new dilemma rule assumption $P \equiv Q$). In this case, a proof that F is a valid formula would be a derivation $\pi : F^+ \Rightarrow \mathsf{R}\big[F \equiv \top\big]$ where R is some (not explicitly contradictory) relation on F. Indeed, if π' is a dilemma proof of F in the sense of definition 4.12, then the derivation

$$\dfrac{\dfrac{F^+}{F^+\big[F \equiv \bot\big] \quad F^+\big[F \equiv \top\big]} \atop \pi'}{F^+\big[F \equiv \top\big]} \tag{4.21}$$

would be a proof of F in the above alternative (and formally more correct) sense.

This example shows that the remark about dilemma not being a natural deduction system in the strict sense is inconsequential. In order to make our description of dilemma agree with that of [49], and in order to get a closer correspondence between the proof system and the proof search algorithm Stålmarck's method, we therefore choose to discuss the dilemma proof system as defined by definitions 4.9 and 4.12.

For the record, we state the following theorem.

Theorem 4.14 (Dilemma is a propositional proof system)
The dilemma proof system as defined by definitions 4.9 and 4.12 is a propositional proof system.

Sketch of proof: In order to prove that dilemma is a propositional proof system in the sense of definition 2.2 on page 11, we need to show that it is a sound and complete "deduction system" (with the reservations in remark 4.13) with polynomial-time checkable proofs.

Assuming that we augment every formula relation in a dilemma derivation with a note about how it was derived from earlier formula relations in the derivation, it is not hard to construct an algorithm that checks dilemma proofs in polynomial time.

To see that dilemma is complete, suppose that F is a tautology and order the n variables in $\mathit{Vars}(F)$ in some order $x_1, \ldots, x_n$. Construct a dilemma derivation which opens with the formula relation $F^\perp$ immediately followed by n nested levels of dilemma rule applications, where the applications on level i branches on the truth value of x_i. In each of the 2^n innermost branches we have one of the 2^n possible different valuations on $\mathit{Vars}(F)$ and the simple rules can be used to propagate values from the variables upwards to subformulas all the way to F. If F is a valid formula we will get contradictions in all of the branches, and closing the dilemma rule applications one by one we finally derive a contradiction at zero-level, showing that F is a tautology.

Dilemma is sound since the simple rules and the dilemma rule are all sound. We leave the detailed proof of this fact to the reader.

The theorem follows. $\square$

Next, we turn our attention to bounds on dilemma proofs and the related notion of *hardness degree* in the dilemma proof system.

4.1.5 Proof Hardness and Bounds on Proof Length

As has already been mentioned, the amount of nested branching needed (or, using the language of definition 4.9, the derivation depth) is a critical factor for the complexity of dilemma proofs. The more branching is needed, the harder a formula relation becomes to refute. Before elaborating on this further, we introduce some terminology.

Definition 4.15 (Hardness degree) *Given $\kappa \in \mathbb{N}$, a formula relation R is said to be* κ-easy *if there is a derivation $\pi : \mathsf{R} \Rightarrow \bot_{\mathsf{R}}$ with $D(\pi) \leq \kappa$ and* κ-hard *if there is no derivation $\pi : \mathsf{R} \Rightarrow \bot_{\mathsf{R}}$ with $D(\pi) < \kappa$. If a formula relation R is both κ-easy and κ-hard, we say that it is* exactly κ-hard *and has* hardness degree $H(\mathsf{R}) = \kappa$.

We extend these definitions to formulas by stipulating that a tautology F is κ-easy, κ-hard or exactly κ-hard (has hardness degree $H(F) = \kappa$) if the corresponding formula relation $F^{\bot}$ is κ-easy, κ-hard or exactly κ-hard (has hardness degree $H\left(F^{\bot}\right) = \kappa$), respectively. For a contradictory formula F, the same measures are defined in terms of the formula relation $F^{\top}$.

For convenience, we adopt the convention that a non-contradictory (and thus non-refutable) formula relation R has hardness degree $H(\mathsf{R}) = \infty$.[3]

The justification for the terminology in definition 4.15 is that the measure of hardness degree of a formula relation actually captures the traditional complexity-inspired notion of hardness in terms of proof size. Easy formulas (i.e. formulas with low hardness degree) have short dilemma proofs while hard formulas (only) have long proofs. Furthermore, when presenting Stålmarck's method, we will see that this algorithm has a favourable asymptotic behaviour with respect to formula size as long as the formulas are easy in the sense of definition 4.15 (which from our point of view serves as the main motivation behind the definition).

In analogy with definitions 2.17 and 2.19, we introduce notation for the minimum length and size of dilemma proofs.

Definition 4.16 *Let R be a contradictory formula relation. Then the minimum length of any dilemma refutation of R is denoted $L_{\mathcal{D}}(\mathsf{R} \vdash \bot)$. The minimum size of any dilemma refutation is denoted $S_{\mathcal{D}}(\mathsf{R} \vdash \bot)$.*

In the same way, we let $L_{\mathcal{D}}(F \vdash \bot)$ and $S_{\mathcal{D}}(F \vdash \bot)$ denote the minimum length and size of a dilemma refutation of a contradictory formula F. If F is a tautology, we use the notation $L_{\mathcal{D}}(\vdash F)$ and $S_{\mathcal{D}}(\vdash F)$ for the minimum length and size of a dilemma proof of F.

The rest of this section is devoted to stating and proving upper and lower bounds on the length (and thus by the inequalities (4.20) on the size) of dilemma derivations π from R in terms of the hardness degree $H(\mathsf{R})$ (theorems 4.18 and 4.23). To simplify the proofs, we first extend the concept of associations to dilemma derivations.

[3] An alternate and perhaps just as natural definition would be that the hardness of a formula relation R is the minimal depth in which R can be proved contradictory *or* satisfiable. In this thesis, however, defining $H(\mathsf{R}) = \infty$ for a non-contradictory relation R suits our purposes better.

Definition 4.17 *Given a dilemma derivation $\pi : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ and an association ψ on R_1, we define π* augmented by ψ, *denoted $\pi[\psi]$ to be the derivation where every relation R in π is augmented with the association ψ. That is, $\pi[\psi]$ is a (possibly non-proper and/or non-standardized) dilemma derivation of $\mathsf{R}_2[\psi]$ from $\mathsf{R}_1[\psi]$.*

For Ψ a set of associations, $\pi[\Psi]$ is defined in an analogous fashion.

We now give a lower bound on the length of dilemma refutations of a formula relation R.

Theorem 4.18 (Lower bound on length of dilemma proofs)
Let R be a contradictory formula relation. Then

$$L_{\mathcal{D}}(\mathsf{R} \vdash \bot) \geq 2^{H(\mathsf{R})/2}.$$

In order to prove the theorem, we introduce a useful piece of notation and show a technical lemma.

By definition 4.15, if a relation R has (finite) hardness degree h there must exist a dilemma derivation $\pi : \mathsf{R} \Rightarrow \bot_{\mathsf{R}}$ of depth h. We write $\mathit{proof}(\mathsf{R})$ to denote some representative (arbitrary but fix) of such a derivation.

Lemma 4.19 (Unrolling lemma)
Suppose that π is a dilemma derivation $\pi : \mathsf{R} \Rightarrow \mathsf{R}[\Psi]$, where $\Psi = \{\psi_1, \ldots, \psi_n\}$ is a set of associations. Set $\Psi'_i = \{\psi_1, \ldots, \psi_{i-1}, {\psi_i}^C\}$ for $1 \leq i \leq n$ and let $h = \max_{1 \leq i \leq n} \{H(\mathsf{R}[\Psi'_i])\}$.

Then $h < \infty$ and there is a derivation $\pi' : \mathsf{R} \Rightarrow \mathsf{R}[\Psi]$ with $D(\pi') \leq h + 1$.

Proof [of lemma 4.19]: Since there is a dilemma derivation $\pi : \mathsf{R} \Rightarrow \mathsf{R}[\Psi]$, the formula relation $\mathsf{R}[\Psi'_i]$ must be (implicitly) contradictory for each i. Indeed, we can augment each relation in the derivation π with the associations Ψ'_i to get a new (possibly non-proper) derivation $\pi[\Psi'_i] : \mathsf{R}[\Psi'_i] \Rightarrow \mathsf{R}[\Psi'_i, \Psi]$. The final relation in $\pi[\Psi'_i]$ is explicitly contradictory (since it contains both ψ_i and ${\psi_i}^C$) and can be extended in one step to the canonical explicitly contradictory formula relation $\bot_{\mathsf{R}}$. If we remove all redundant derivation steps from this derivation, we get a proper dilemma refutation of $\mathsf{R}[\Psi'_i]$, which we may call π'_i. We see that $H(\mathsf{R}[\Psi'_i])$ is finite for all i and consequently $h < \infty$.

By the above paragraph, for each formula relation $\mathsf{R}[\Psi'_i]$ there is a dilemma refutation $\mathit{proof}(\mathsf{R}[\Psi'_i])$ with depth $H(\mathsf{R}[\Psi'_i])$. We use these refutations as building blocks in the derivation presented in figure 4.1 on the next page, which we call $\mathit{unroll}(\mathsf{R}, \Psi)$. For all i it holds that $D(\mathit{proof}(\mathsf{R}[\Psi'_i])) \leq h$, so $D(\mathit{unroll}(\mathsf{R}, \Psi)) \leq h + 1$ and the lemma follows. □

Remark 4.20 Note that the construction in lemma 4.19 works equally well if we start from a dilemma derivation $\pi : \mathsf{R} \Rightarrow \mathsf{R}[\Psi, \Phi]$, where Φ is an arbitrary set of associations. The resulting derivation $\mathit{unroll}(\mathsf{R}, \Psi)$ will still be from R to $\mathsf{R}[\Psi]$, though.

Proof [of theorem 4.18]: We prove the theorem by showing the contrapositive: If the length $L(\pi)$ of a (proper) dilemma derivation $\pi : \mathsf{R} \Rightarrow \bot_{\mathsf{R}}$ is

$$
\begin{array}{c}
\mathsf{R} \\ \hline
\mathit{proof}\left(\mathsf{R}\left[{\psi_1}^C\right]\right) \qquad\qquad \mathsf{R}\left[\psi_1\right] \\ \hline
\mathsf{R}\left[\psi_1\right] \\ \hline
\mathit{proof}\left(\mathsf{R}\left[\psi_1, {\psi_2}^C\right]\right) \qquad\qquad \mathsf{R}\left[\psi_1, \psi_2\right] \\ \hline
\mathsf{R}\left[\psi_1, \psi_2\right] \\
\vdots \\
\mathsf{R}\left[\psi_1, \ldots, \psi_{n-1}\right] \\ \hline
\mathit{proof}\left(\mathsf{R}\left[\psi_1, \ldots, \psi_{n-1}, {\psi_n}^C\right]\right) \quad \mathsf{R}\left[\psi_1, \ldots, \psi_{n-1}, \psi_n\right] \\ \hline
\mathsf{R}\left[\Psi\right]
\end{array}
$$

Figure 4.1: $\mathit{unroll}(\mathsf{R}, \Psi)$ in the proof of lemma 4.19.

strictly less than b, then the hardness $H(\mathsf{R})$ of the relation R must be strictly less than $2 \log b$. The proof is by (strong) induction on b.

Base case: If $b = 2$ then $L(\pi) = 1$, i.e. π consists of a single (explicitly contradictory) formula relation R and trivially $H(\mathsf{R}) = 0$.

Induction step: Assume that for all $b' < b$ and derivations $\pi' : \mathsf{R}' \Rightarrow \bot_{\mathsf{R}'}$ it holds that $L(\pi') < b'$ implies $H(\mathsf{R}') < 2 \log b'$. Let $\pi : \mathsf{R} \Rightarrow \bot_{\mathsf{R}}$ be a derivation of length $L(\pi) < b$.

The derivation π starts either with a simple derivation or with an application of the dilemma rule. We consider the two cases in turn.

1. Suppose that $\pi = \pi_1 \bullet \pi_2$ where $\pi_1 : \mathsf{R} \Rightarrow \mathsf{R}_1$ is a simple derivation and $\pi_2 : \mathsf{R}_1 \Rightarrow \bot_{\mathsf{R}}$ is a (general) dilemma derivation. We have $L(\pi_2) = b - 1$, so by the induction hypotesis $D(\pi_2) < 2 \log(b-1) < 2 \log b$, and obviously $D(\pi_1) = 0$. Thus $D(\pi) < 2 \log b$.

2. If π starts by a dilemma rule application, it has the form

$$
\begin{array}{c}
\mathsf{R} \\ \hline
\pi_1 \quad \pi_2 \\ \hline
\pi_3
\end{array}
\tag{4.22}
$$

for derivations $\pi_1 : \mathsf{R}[\phi] \Rightarrow \mathsf{R}_1$, $\pi_2 : \mathsf{R}[\phi^C] \Rightarrow \mathsf{R}_2$ and $\pi_3 : \mathsf{R}_1 \sqcap \mathsf{R}_2 \Rightarrow \bot_{\mathsf{R}}$ (where ϕ is some association on R). To simplify the applications below of lemma 4.19, let Ψ be a set of associations chosen so that $\mathsf{R}[\Psi] = \mathsf{R}_1 \sqcap \mathsf{R}_2$. Then π_3 is a refutation of $\mathsf{R}[\Psi]$, and the derivations π_1 and π_2 can be written as $\pi_1 : \mathsf{R}[\phi] \Rightarrow \mathsf{R}[\phi, \Psi, \Phi_1]$ and $\pi_2 : \mathsf{R}[\phi^C] \Rightarrow \mathsf{R}[\phi^C, \Psi, \Phi_2]$ for some sets of associations Φ_1 and Φ_2.

Now since $L(\pi) = L(\pi_1) + L(\pi_2) + L(\pi_3) + 1 < b$ (and $b \geq 3$), it must hold that at least two of the π_i are shorter than $b/2 - 1$. We get three cases:

(a) $L(\pi_1) < b/2 - 1$ and $L(\pi_2) < b/2 - 1$,

(b) $L(\pi_1) < b/2 - 1$ and $L(\pi_3) < b/2 - 1$,

(c) $L(\pi_2) < b/2 - 1$ and $L(\pi_3) < b/2 - 1$.

(a) The derivations π_1 and π_2 are both short, so we apply the induction hypotesis and lemma 4.19 (with remark 4.20) to construct equivalent shallow derivations in both dilemma rule branches.

Starting with $\pi_1 : \mathsf{R}[\phi] \Rightarrow \mathsf{R}[\phi, \Psi, \Phi_1]$, if $L(\pi_1) < b/2 - 1$ it follows that the refutations $\pi'_i : \mathsf{R}[\phi, \Psi'_i] \Rightarrow \bot_{\mathsf{R}}$ constructed in the proof of lemma 4.19 must be shorter than $b/2$. But then by the induction hypotesis each of the relations $\mathsf{R}[\phi, \Psi'_i]$ has hardness degree less than $2\log(b/2) = 2\log b - 2$. Thus, the derivation $\mathit{unroll}(\mathsf{R}[\phi], \Psi)$ constructed as in figure 4.1 has depth strictly less than $2\log b - 1$.

In the same way, applying the unrolling lemma to π_2 we see that there is a derivation $\mathit{unroll}(\mathsf{R}[\phi^C], \Psi)$ from $\mathsf{R}[\phi^C]$ to $\mathsf{R}[\phi^C, \Psi]$ in depth less than $2\log b - 1$.

Finally, $L(\pi_3) < b - 1$ so by the induction hypotesis it must hold that $H(\mathsf{R}[\Psi]) < 2\log(b-1) < 2\log b$.

Putting it all together, we arrive at the dilemma proof

$$\frac{\dfrac{\mathsf{R}}{\mathit{unroll}(\mathsf{R}[\phi], \Psi) \quad \mathit{unroll}(\mathsf{R}[\phi^C], \Psi)}}{\mathit{proof}(\mathsf{R}[\Psi])} \tag{4.23}$$

of depth strictly less than $2\log b$, so $H(\mathsf{R}) < 2\log b$.

(b) In this case π_1 and π_3 are short, but π_2 is not. Therefore, we insert shallow modifications of π_1 and π_3 deriving $\bot$ into the left branch and move π_2 below the dilemma rule application.

Just as in case (a), we can construct a derivation $\mathit{unroll}(\mathsf{R}[\phi], \Psi)$ from $\mathsf{R}[\phi]$ to $\mathsf{R}[\phi, \Psi]$ in depth less than $2\log b - 2$. Then we take $\pi_3 : \mathsf{R}[\Psi] \Rightarrow \bot_{\mathsf{R}}$ and augment this derivation with the association ϕ. The resulting refutation $\pi_3[\phi]$ of $\mathsf{R}[\phi, \Psi]$ has length less than $b/2$, so by the induction hypotesis $H(\mathsf{R}[\phi, \Psi]) < 2\log b - 2$. Inserting $\mathit{unroll}(\mathsf{R}[\phi], \Psi)$ followed by $\mathit{proof}(\mathsf{R}[\phi, \Psi])$ into the left branch of the dilemma rule application, we get a dilemma derivation of $\mathsf{R}[\phi^C]$ in depth less than $2\log b - 1$.

We now use π_2 to derive $\mathsf{R}[\phi^C, \Psi, \Phi_2]$ from $\mathsf{R}[\phi^C]$ and then augment π_3 a second time to get the refutation $\pi_3[\phi^C, \Phi_2]$ of $\mathsf{R}[\phi^C, \Psi, \Phi_2]$. The composition of these two derivations is a dilemma proof from $\mathsf{R}[\phi^C]$ in length less than $b-1$, and by yet another appeal to the induction hypotesis we conclude that there is a refutation $\mathit{proof}(\mathsf{R}[\phi^C])$ with depth less than $2\log b$.

Combining the derivations above we get a refutation of R

$$\frac{\dfrac{\mathsf{R}}{\begin{array}{ll}\mathit{unroll}(\mathsf{R}[\phi], \Psi) & \mathsf{R}[\phi^C] \\ \mathit{proof}(\mathsf{R}[\phi, \Psi]) & \end{array}}}{\mathit{proof}(\mathsf{R}[\phi^C])} \tag{4.24}$$

in depth less than $2\log b$, which shows that $H(\mathsf{R}) < 2\log b$.

(c) Symmetric to case (b).

The theorem follows by the induction principle. □

When comparing dilemma to other proof systems, it can be convenient to express the lower bound on the length of dilemma proofs in terms of formulas instead of relations.

Corollary 4.21 (Lower bound, formula version)
Let F be a valid (or contradictory) formula and suppose that π is a dilemma proof (or refutation, respectively) of F. Then $L(\pi) \geq 2^{H(F)/2}$.

Formula relations respect complement, so if one equivalence class in a relation R contains a particular set of subformulas, there is a complementary equivalence class in R containing the complements of those subformulas. Each successful application of a simple rule to R merges a pair of equivalence classes as well as the corresponding pair of complementary classes, and so reduces the number of equivalence classes in R by two. Non-redundant applications of the dilemma rule certainly reduce the number of equivalence classes by at least two. For this reason a proper dilemma derivation cannot go on forever. This simple observation underlies the proof of the upper bound on the length of dilemma proofs which we shall give presently.

Definition 4.22 (Proper d-derivation) *A* proper d-derivation (d-refutation, d-proof) *is a dilemma derivation (refutation, proof) in depth d without redundant rule applications.*

Given a fix d, the theorem below gives a bound for the length of a proper d-derivation from a formula relation R which is polynomial in the size $|\mathsf{R}|$ of the relation, i.e. the number of equivalence classes in R. (To be more precise, the bound is in $|\mathsf{R}|/2$, which in view of what has been said above can be considered to be the number of "independent" equivalence classes of R.)

Theorem 4.23 (Upper bound on length of proper d-derivations)
Let $f(d,s)$ be defined by

$$f(d,s) = \begin{cases} s & \text{for } d = 0 \text{ or } s \leq 3, \\ s + 2 \cdot \sum_{j=1}^{s-1} f(d-1,j) & \text{for } d > 0 \text{ and } s > 3. \end{cases}$$

Suppose that R and R' are compatible formula relations (where R is not explicitly contradictory) and set $s := |\mathsf{R}|/2$.

If $\pi : \mathsf{R} \Rightarrow \mathsf{R}'$ is a proper d-derivation, then $L(\pi) \leq f(d,s)$. In particular, it holds that $L(\pi) \leq s^{d+1}$.

Proof: By induction on d.

Base case ($d = 0$): π is a 0-derivation in which only simple rules are applied, and so is just a sequence of formula relations. By the reasoning above, for each new relation in the sequence the number of equivalence classes in the relation is decreased by (at least) two. In other words, $|\mathsf{R}| - |\mathsf{R}'| \geq 2 \cdot (L(\pi) - 1)$, from which it follows that $L(\pi) \leq |\mathsf{R}|/2 = s$.

Induction step: Assume that the bound holds for all $d' < d$ and let π be a d-derivation. π consists of at most $\lfloor (|\mathsf{R}| - |\mathsf{R}'|)/2 \rfloor \leq s - 1$ applications of simple or dilemma rules. Clearly, the worst case is when π is a composition of

$s-1$ dilemma rule applications

$$\begin{array}{c} \mathsf{R} \\ \hline \rho_1 \qquad \sigma_1 \\ \hline \mathsf{R}_1 \\ \hline \rho_2 \qquad \sigma_2 \\ \hline \mathsf{R}_2 \\ \vdots \\ \mathsf{R}_{s-2} \\ \hline \rho_{s-1} \quad \sigma_{s-1} \\ \hline \mathsf{R}' \end{array} \tag{4.25}$$

where the branches ρ_j, σ_j must have depth less than d. The first relation in each ρ_j and σ_j contains at most $2(s-j)$ equivalence classes. By the induction hypotesis we have $L(\rho_j) \leq f\big(d-1, s-j\big)$ and $L(\sigma_j) \leq f\big(d-1, s-j\big)$. For the length $L(\pi)$ of the whole derivation we obtain

$$L(\pi) = s + \sum_{j=1}^{s-1} L(\rho_j) + \sum_{j=1}^{s-1} L(\sigma_j) \leq s + 2 \cdot \sum_{j=1}^{s-1} f\big(d-1, j\big) = f\big(d, s\big), \tag{4.26}$$

so the bound holds for d as well.

By the induction principle, $L(\pi) \leq f(d,s)$ holds for all $d \geq 0$.

It is easy to show that $f(d,s) \leq s^{d+1}$, so the second bound is just a weaker version of the first. The theorem follows. □

For reference, we state the following reformulation of the second part of theorem 4.23 for dilemma *proofs* as a corollary.

Corollary 4.24 (Upper bound on length of dilemma proofs)
Let R *be a contradictory formula relation with hardness degree* $H(\mathsf{R}) \leq h$ *and suppose that* $\pi : \mathsf{R} \Rightarrow \perp_{\mathsf{R}}$ *is a proper* h*-refutation of* R*. Then*

$$L(\pi) \leq \big(|\mathsf{R}|/2\big)^{h+1}$$

and it follows that

$$L_{\mathcal{D}}(\mathsf{R} \vdash \perp) \leq \mathrm{O}\left(|\mathsf{R}|^{H(\mathsf{R})+1}\right).$$

Since the size of any formula relation R on a formula F is bounded by $|\mathsf{R}| \leq 2 \cdot (S(F) + 1)$, the bound on the minimum length of a dilemma proof or refutation of a formula F can be expressed in terms of the size $S(F)$ of the formula as in the next corollary.

Corollary 4.25 (Upper bound on length of proofs, formula version)
Let F *be a valid (or contradictory) formula and suppose that* π *is any dilemma proof (or refutation, respectively) of* F *in minimal depth* $H(F)$*. Then*

$$L(\pi) \leq \mathrm{O}\left(S(F)^{H(F)+1}\right).$$

Suppose that F is a tautological formula with hardness degree $H(F) = h$. From corollaries 4.21 and 4.25 it follows that we can search exhaustively for a proof of F in depth h, knowing that the proof eventually found will be at most quasi-polynomially larger than a smallest possible proof. This observation lies at heart of Stålmarck's proof search algorithm, which will be described in section 4.2. We conclude our discussion on proof bounds by stating this as a theorem and proving it.

Theorem 4.26
Let F be a tautology having hardness degree $H(F) = h$. Then the size and length of any proper h-proof of F is quasi-polynomial in the size $S_{\mathcal{D}}(\vdash F)$ of a smallest possible proof of F.

Proof: Let π be an arbitrary proper h-proof of F. It is sufficient to show that the length $L(\pi)$ is quasi-polynomial in $S_{\mathcal{D}}(\vdash F)$, since by the inequalities (4.20) it holds that $S(\pi) \leq L(\pi) \cdot \mathrm{O}(S(F)^2)$. Note that $L(\pi) \leq \mathrm{O}(S(F))^{h+1}$ by corollary 4.25.

If F is 0-easy, $L(\pi) \leq \mathrm{O}(S(F)) \leq S_{\mathcal{D}}(\vdash F)$, where the last inequality follows since $S_{\mathcal{D}}(\vdash F) \geq S(F)$ (each formula relation in π contains a copy of F).

Now suppose $h \geq 1$. By corollary 4.21 we have $S_{\mathcal{D}}(\vdash F) \geq L_{\mathcal{D}}(\vdash F) \geq 2^{h/2}$. Using this inequality as well as $S_{\mathcal{D}}(\vdash F) \geq S(F)$ we get

$$L(\pi) \leq \mathrm{O}\left(S(F)^{h+1}\right) \leq \left(\left(2^{h/2}\right)^4\right)^{\mathrm{O}(\log S(F))} \leq S_{\mathcal{D}}(\vdash F)^{\mathrm{O}(\log S_{\mathcal{D}}(\vdash F))}, \quad (4.27)$$

so $L(\pi)$ is quasi-polynomial in $S_{\mathcal{D}}(\vdash F)$ and the theorem follows. □

4.1.6 The Hardness Degree Hierarchy

The corollaries of the previous section give lower and upper bounds on proof length in terms of hardness of individual formulas. We can translate these bounds into bounds for sequences as follows.

Suppose that $(F_1, F_2, F_3, \ldots)$ is a sequence of formulas with formula sizes $S(F_n)$ polynomial in n. If the hardness degrees $H(F_n)$ increase linearly with n, the sequence requires exponentially increasing proof lengths (and sizes) by corollary 4.21. By corollary 4.25, if the hardness grows logarithmically the proof lengths and sizes increase (at most) quasi-polynomially, and for constant hardness degrees we get polynomial-size proofs. Thus our concept of hardness degree really captures the more traditional proof-theoretical measure of hardness in terms of proof size.

The question remains, however, whether the hardness degrees actually form an infinite hierarchy or whether everything collapses at some particular degree d. The answer to this question is that the infinite hierarchy does indeed exist. Ajtai [1] has shown that there cannot exist polynomial-size proofs of the so called *PHP*-formulas, which encode the pigeonhole principle, in bounded depth Frege systems. Furthermore, according to [51] it is easy to polynomially embed dilemma proofs of depth d of the *PHP*-formulas in a bounded depth Frege system. From the upper bound in corollary 4.25 it follows that the hardness of these formulas must be increasing. (See also remark 6.24 on page 113.)

Perhaps it is not very surprising that the hierarchy is infinite. If it collapsed at some level d, then all tautologies in propositional logic would be d-easy and so by corollary 4.25 would have polynomial-size dilemma proofs. In other words, dilemma would be a p-bounded propositional proof system, which by theorem 2.15 would imply that $\mathsf{NP} = \mathsf{co}\text{-}\mathsf{NP}$. This is an equality which would go contrary to the intuition of most complexity theorists.

It should be noted, though, that the existence of the hardness degree hierarchy depends on the fact that dilemma is an analytic proof system, i.e. that it respects the subformula principle (although this has not been stated explicitly, it is implicit in the definitions 4.9 and 4.12). If this restriction is removed the hierarchy breaks down. We show this (in a rather informal manner) in the following example.

Example 4.1 Consider the proof π in figure 4.2 on the next page (where for simplicity of notation we have replaced all but the first and final formula relations with just the new association added to the relation at the corresponding step in the derivation). Assume that $D(\pi_1) = d - 1$ and $D(\pi_i) = d - 2$ for $2 \leq i \leq 6$, so that it holds that $D(\pi) = d$.

If we are allowed to ignore the subformula principle (and extend the definition of formula relation to include arbitrary formulas), we can use the rules for bi-implication in figure A.5 on page 129 to move freely between associations $P \equiv Q$ and $P \leftrightarrow Q \equiv \top$ or associations $P \equiv Q^C$ and $P \leftrightarrow Q \equiv \bot$. Thus, we can transform all dilemma rule applications branching over the equivalence of P and Q to applications branching over the truth or falsehood of $P \leftrightarrow Q$. Furthermore, we can lump together the dilemma rule assumptions $P \leftrightarrow Q \equiv \top$ and $R \leftrightarrow S \equiv \top$ to $(P \leftrightarrow Q) \wedge (R \leftrightarrow S) \equiv \top$, and then so to speak deal with the assumptions in this conjunction in turn, thereby decreasing the depth of the proof.

By applying these tricks, we arrive at the "dilemma proof" π' in figure 4.3 in depth $D(\pi') = d - 1$. Unless all of our newly constructed formulas in the dilemma rule assumptions happen to be subformulas of F, this proof is *non-analytic*, i.e. it does not respect the subformula principle. $\diamond$

The above example outlines how the depth of a dilemma proof can be reduced by one when discarding the subformula principle. It is not hard to see that the same techniques can be generalized to transform dilemma proofs of any depth to equivalent non-analytic proofs of depth one. We omit the details.

4.2 The Proof Method

In the previous section we defined the dilemma proof system and investigated it in some detail. We now turn to the question how to construct an efficient proof search algorithm based on this system. The construction is divided into three steps:

- First, we devise good data structures for representing formulas, formula relations and derivations.
- Then, we construct an efficient algorithm that searches exhaustively for (preferably short) dilemma proofs.
- Finally, we add some optimization to our algorithm to make it run fast not only asymptotically but also in practice.

$$\begin{array}{c} F^{\top} \\ \hline \begin{array}{cc} \begin{array}{c} P \equiv Q \\ \pi_2 \\ \hline \begin{array}{cc} R \equiv S & R \equiv S^C \\ \pi_3 & \pi_4 \end{array} \\ \hline \pi_5 \end{array} & \begin{array}{c} P \equiv Q^C \\ \pi_1 \\ \\ \\ \\ \end{array} \end{array} \\ \hline \pi_6 \\ \bot_{F+} \end{array}$$

Figure 4.2: Schematic dilemma proof π respecting the subformula principle.

$$\begin{array}{c} F^{\top} \\ \hline \begin{array}{cc} \begin{array}{c} (P \leftrightarrow Q) \wedge (R \leftrightarrow S) \equiv \top \\ P \leftrightarrow Q \equiv \top \\ R \leftrightarrow S \equiv \top \\ P \equiv Q \\ R \equiv S \\ \pi_2 \\ \pi_3 \\ \pi_5 \\ \pi_6 \\ \bot_{F+} \end{array} & \begin{array}{c} (P \leftrightarrow Q) \wedge (R \leftrightarrow S) \equiv \bot \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \end{array} \\ \hline (P \leftrightarrow Q) \wedge (R \leftrightarrow S) \equiv \bot \\ \hline \begin{array}{cc} \begin{array}{c} P \leftrightarrow Q \equiv \top \\ R \leftrightarrow S \equiv \bot \\ P \equiv Q \\ R \equiv S^C \\ \pi_2 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \bot_{F+} \end{array} & \begin{array}{c} P \leftrightarrow Q \equiv \bot \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \end{array} \\ \hline P \leftrightarrow Q \equiv \bot \\ P \equiv Q^C \\ \pi_1 \\ \pi_6 \\ \bot_{F+} \end{array}$$

Figure 4.3: Transformed non-analytic proof π'.

The algorithm discussed in this section is called Stålmarck's method. The main motivation for this algorithm is theorem 4.26, which says that we can expect to find reasonably small proofs of a formula F (in terms of the minimum size of any dilemma proof of F) by searching exhaustively for *shallow* proofs, i.e. proofs with small branching depth.

We describe Stålmarck's method by attending to the above three steps in turn. In section 4.2.1 we introduce *triplets* and try to develop a feel for dilemma derivations by giving a few examples. At the end of the section we introduce some new terminology inspired by these examples. In section 4.2.2, we introduce the κ-saturation algorithm on which Stålmarck's method is based and analyze the running times of (different versions of) κ-saturation and Stålmarck's method. We conclude by giving a very brief overview in section 4.2.3 of some of the additional optimizations and features in the commercial proof engine Prover Plug-in™, which implements an extended version of Stålmarck's method. Needless to say, a more detailed study of what tricks are used when making the Stålmarck method into a software product goes far beyond our interests in this thesis.

As in the previous section, our presentation is based on [49], but the material has been greatly expanded and elaborated. The terminology in section 4.2.2 is inspired partly by [58], although the definitions there differ quite substantially from ours ([58] defines the terms more abstractly, without any reference to dilemma derivations, and uses sets of *judgements* instead of formula relations). Section 4.2.3 makes use of material from [44].

4.2.1 Data Structures and Derivation Fundamenta

As was noted in section 4.1.4, representing a formula relation by writing out all subformulas in it in full is very inefficient indeed. A more economical way of representing formula relations is to label all (compound) subformulas and use these labels instead. Following [44, 49], we call these labels *triplets*.

Definition 4.27 (Triplet) *For F a formula in propositional logic, the* triplet variables *U_i of F are defined by introducing variables U_P for all subformulas $P \doteq Q \circ R$ in F (where $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$) and setting $U_x := x$ for variables $x \in Vars(F)$ and $U_P := \neg U_{P'}$ for subformulas $P \doteq \neg P'$ in F.*

If $P \doteq Q \circ R$ is a subformula in F, we say that $U_P : U_Q \circ U_R$ is the triplet *representing $P \doteq Q \circ R$, where U_Q and U_R are called the* (triplet) children *of U_P and U_P is called the* (triplet) parent *of U_Q and U_R. The set $\{U_P \mid P \in Sub(F)\}$ is the* triplet representation *of F.*

Using triplets, formula relations on F can be represented as relations on the set of triplet variables of F, with identical subformulas in F represented by the same triplet. It is not too hard to see that the number of symbols in a triplet representation of a formula F is at most $\mathrm{O}(S(F))$ and at least $\Omega(\log S(F))$ (the latter bound follows for instance by induction). Any formula relation on F can be represented in $\mathrm{O}(S(F))$ number of symbols, since the number of triplets is $\mathrm{O}(S(F))$.

While the triplet representation is certainly much more efficient than the naive $\mathrm{O}\big(S(F)^2\big)$-representation used in the formal definition of the dilemma proof system, and thus is crucial when implementing proof search algorithms, an extra factor $S(F)$ does not influence the theoretical results about dilemma in any significant way. We will therefore use triplet representation when describing the Stålmarck proof search algorithm, but stick to the less economical but simpler subformula representation for the theoretical analysis.

For a further optimization of formula relation representation, note that if C is an equivalence class of a formula relation R, then so is C^C. Therefore, in implementations of proof search algorithms based on dilemma we can save memory by storing only half of the equivalence classes and making the complementary equivalence classes implicit. When two equivalence classes are merged, their "shadow" complementary classes are also merged implicitly.

In the rest of this section, we work through some examples of dilemma derivations in triplet representation to get a feel for the proof system. On the basis of these examples we then introduce some terminology which we will need when describing Stålmarck's method and the κ-saturation algorithm in terms of which it is defined.

The propagation rules used in the derivations are presented in triplet format in figure 4.4 on the next page.

Example 4.2 (0-depth proof) The formula

$$F_1 := (x \vee y) \wedge (x \vee z) \to x \vee (y \wedge z)$$

is a tautology. It is represented by the following set of triplets:

$$\begin{array}{lll} U_1 : x \vee y & U_3 : U_1 \wedge U_2 & U_5 : x \vee U_4 \\ U_2 : x \vee z & U_4 : y \wedge z & U_6 : U_3 \to U_5 \end{array}$$

A dilemma proof of F_1 (where we display only half of the equivalence classes as explained above) is:

$$\begin{array}{ll}
\{[\top, \neg U_6], [x], [y], [z], [U_1], [U_2], [U_3], [U_4], [U_5]\} & [{F_1}^{\perp}] \\
\{[\top, U_3, \neg U_6], [x], [y], [z], [U_1], [U_2], [U_4], [U_5]\} & [\text{rule (4.30a) on } U_6] \\
\{[\top, U_3, \neg U_5, \neg U_6], [x], [y], [z], [U_1], [U_2], [U_4]\} & [\text{rule (4.30b) on } U_6] \\
\{[\top, U_1, U_3, \neg U_5, \neg U_6], [x], [y], [z], [U_2], [U_4]\} & [\text{rule (4.28a) on } U_3] \\
\{[\top, U_1, U_2, U_3, \neg U_5, \neg U_6], [x], [y], [z], [U_4]\} & [\text{rule (4.28a) on } U_3] \\
\{[\top, \neg x, U_1, U_2, U_3, \neg U_5, \neg U_6], [y], [z], [U_4]\} & [\text{rule (4.29a) on } U_5] \\
\{[\top, \neg x, U_1, U_2, U_3, \neg U_4, \neg U_5, \neg U_6], [y], [z]\} & [\text{rule (4.29a) on } U_5] \\
\{[\top, \neg x, y, U_1, U_2, U_3, \neg U_4, \neg U_5, \neg U_6], [z]\} & [\text{rule (4.29b) on } U_1] \\
\{[\top, \neg x, y, z, U_1, U_2, U_3, \neg U_4, \neg U_5, \neg U_6]\} & [\text{rule (4.29b) on } U_2] \\
\{[\top, \neg x, y, z, U_1, U_2, U_3, \mathbf{U_4}, \neg \mathbf{U_4}, \neg U_5, \neg U_6]\} & [\text{rule (4.28b) on } U_4] \\
\{[\top, \perp, x, \neg x, y, \neg y, z, \neg z, U_1, \neg U_1, \ldots, U_6, \neg U_6]\} & [\perp_{F_1^+} \text{ by rule (4.18)}]
\end{array}$$

Since this proof uses only simple rules, F_1 is 0-easy. ◇

If we turn the formula in example 4.2 around, we get the formula

$$F := x \vee (y \wedge z) \to (x \vee y) \wedge (x \vee z). \tag{4.32}$$

Rules (4.30a) and (4.30b) applied on the outermost implication of F yield that $x \vee (y \wedge z) \equiv \top$ and $(x \vee y) \wedge (x \vee z) \equiv \perp$, but there are no simple rules for conjunction or disjunction that can propagate these values any further. It follows

$$\frac{U \equiv \top}{V \equiv \top} \qquad \frac{U \equiv \top}{W \equiv \top} \qquad (U : V \wedge W) \tag{4.28a}$$

$$\frac{V \equiv \top}{U \equiv W} \qquad \frac{W \equiv \top}{U \equiv V} \qquad (U : V \wedge W) \tag{4.28b}$$

$$\frac{U \equiv \bot}{V \equiv \bot} \qquad \frac{U \equiv \bot}{W \equiv \bot} \qquad (U : V \vee W) \tag{4.29a}$$

$$\frac{V \equiv \bot}{U \equiv W} \qquad \frac{W \equiv \bot}{U \equiv V} \qquad (U : V \vee W) \tag{4.29b}$$

$$\frac{U \equiv \bot}{V \equiv \top} \qquad (U : V \to W) \tag{4.30a}$$

$$\frac{U \equiv \bot}{W \equiv \bot} \qquad (U : V \to W) \tag{4.30b}$$

$$\frac{U \equiv W^C}{V \equiv \bot} \qquad (U : V \to W) \tag{4.30c}$$

$$\frac{V \equiv \bot}{U \equiv \top} \qquad (U : V \to W) \tag{4.30d}$$

$$\frac{W \equiv \bot}{U \equiv V^C} \qquad (U : V \to W) \tag{4.30e}$$

$$\frac{V \equiv W^C}{U \equiv V^C} \qquad (U : V \to W) \tag{4.30f}$$

$$\frac{U \equiv \top}{V \equiv W} \qquad (U : V \leftrightarrow W) \tag{4.31a}$$

$$\frac{U \equiv \bot}{V \equiv W^C} \qquad (U : V \leftrightarrow W) \tag{4.31b}$$

$$\frac{U \equiv V}{W \equiv \top} \qquad \frac{U \equiv W}{V \equiv \top} \qquad (U : V \leftrightarrow W) \tag{4.31c}$$

$$\frac{U \equiv V^C}{W \equiv \bot} \qquad \frac{U \equiv W^C}{V \equiv \bot} \qquad (U : V \leftrightarrow W) \tag{4.31d}$$

$$\frac{V \equiv \top}{U \equiv W} \qquad \frac{W \equiv \top}{U \equiv V} \qquad (U : V \leftrightarrow W) \tag{4.31e}$$

$$\frac{V \equiv \bot}{U \equiv W^C} \qquad \frac{W \equiv \bot}{U \equiv V^C} \qquad (U : V \leftrightarrow W) \tag{4.31f}$$

$$\frac{V \equiv W}{U \equiv \top} \qquad (U : V \leftrightarrow W) \tag{4.31g}$$

$$\frac{V \equiv W^C}{U \equiv \bot} \qquad (U : V \leftrightarrow W) \tag{4.31h}$$

Figure 4.4: Examples of propagation rules in triplet format.

that F is 1-hard. Branching on for example the truth value of x gives a 1-depth proof, so F is 1-easy and $H(F_2) = 1$.

We choose not to elaborate on the details. Instead, we give another example of a exactly 1-hard formula, which is more interesting in that it hints at some of the power of the dilemma propagation rules.

Example 4.3 (1-depth proof) Consider the formula

$$F_2 := \ ((x \leftrightarrow y) \leftrightarrow x) \leftrightarrow y$$

represented by the triplets

$$U_1 \ : x \leftrightarrow y \qquad U_2 \ : U_1 \leftrightarrow x \qquad U_3 \ : U_2 \leftrightarrow y.$$

If we assume that F_2 is false and propagate, we get:

$$\begin{array}{ll} \{[\top, \neg U_3], [x], [y], [U_1], [U_2]\} & [F_2{}^{\perp}] \\ \{[\top, \neg U_3], [x], [y, \neg U_2], [U_1]\} & \text{[rule (4.31b) on } U_3] \end{array}$$

After this first derivation step we are stuck, since no further consequences can be derived by simple rules.

To convince ourselves of this, we need only investigate the equivalence classes of cardinality greater than one. If we study the dilemma proof system simple rules in figures A.2, A.3, A.4 and A.5, we see that a simple rule can be applied on a triplet $U \ : V \circ W$ only if either one of the triplet variables U, V and W is a member of a determinate equivalence class (which in addition to the triplet variable contains $\top$ or $\perp$) or if two of them are in the same (indeterminate) equivalence class. So in order to decide whether any further consequences can be derived from a relation R or not, it is sufficient to investigate triplets and parents of triplets contained in equivalence classes C of R of cardinality $|\mathsf{C}| \geq 2$.

In the case of F_2, the candidate equivalence classes are $\mathsf{C}_1 = [\top, \neg U_3]$ and $\mathsf{C}_2 = [y, \neg U_2]$. For the *TRUE*-class C_1, since U_3 has no parents the only simple rule applicable is rule (4.31b), which has already been applied. Turning to the indeterminate class C_2, the only possibility is that y and U_2 have a common parent. This is indeed the case, but the rule (4.31h) is just the inverse of rule (4.31b) and yields no new associations. By the preceding paragraph, it follows that no other simple rules apply. When this is the case we say that a formula relation is *0-saturated*. (We return to the concept of saturation and give a more formal definition in section 4.2.2.)

Thus, the formula F_2 is 1-hard, but it can be proved a tautology by making a dilemma rule derivation branching on y. In the branch assuming $y \equiv \perp$ we get the derivation:

$$\begin{array}{ll} \{[\top, \neg y, U_2, \neg U_3], [x], [U_1]\} & [y \equiv \perp] \\ \{[\top, \neg y, U_2, \neg U_3], [x, U_1]\} & \text{[rule (4.31a) on } U_2] \\ \{[\top, \mathbf{y}, \neg \mathbf{y}, U_2, \neg U_3], [x, U_1]\} & \text{[rule (4.31c) on } U_1] \\ \{[\top, \perp, x, \neg x, y, \neg y, \ldots, U_3, \neg U_3]\} & [\perp_{F_2{}^+} \text{ by rule (4.18)]} \end{array}$$

and in the other branch, where $y \equiv \top$, we derive:

$$\begin{array}{ll} \{[\top, y, \neg U_2, \neg U_3], [x], [U_1]\} & [y \equiv \top] \\ \{[\top, y, \neg U_2, \neg U_3], [x, \neg U_1]\} & \text{[rule (4.31b) on } U_2] \\ \{[\top, \mathbf{y}, \neg \mathbf{y}, \neg U_2, \neg U_3], [x, \neg U_1]\} & \text{[rule (4.31d) on } U_1] \\ \{[\top, \perp, x, \neg x, y, \neg y, \ldots, U_3, \neg U_3]\} & [\perp_{F_2{}^+} \text{ by rule (4.18)]} \end{array}$$

(note the applications of rules (4.31c) and (4.31d), which deduce the truth value of y from the equivalence of x and U_1 although their truth values are not known). We arrive at contradictions in both branches, so closing the dilemma rule application yields a contradiction.

Consequently, F_2 is a exactly 1-hard tautology. $\diamond$

Theorem 4.14 tells us that dilemma is complete, i.e. that all tautological formulas are provable, but what happens if we try to prove a formula which is *not* a tautology?

Example 4.4 (Counter-model) Let us change the connective in F_2 to implication and try to prove

$$F_3 := \ ((x \to y) \to x) \to y.$$

We represent F_3 by the set of triplets

$$U_1 \ : x \to y \qquad U_2 \ : U_1 \to x \qquad U_3 \ : U_2 \to y.$$

As before, we assume that the formula is false and propagate:

$$\begin{array}{ll}
\{[\top, \neg U_3], [x], [y], [U_1], [U_2]\} & [{F_3}^{\perp}] \\
\{[\top, U_2, \neg U_3], [x], [y], [U_1]\} & [\text{rule (4.30a) on } U_3] \\
\{[\top, \neg y, U_2, \neg U_3], [x], [U_1]\} & [\text{rule (4.30b) on } U_3] \\
\{[\top, \neg y, U_2, \neg U_3], [x, \neg U_1]\} & [\text{rule (4.30e) on } U_1] \\
\{[\top, x, \neg y, \neg U_1, U_2, \neg U_3]\} & [\text{rule (4.30f) on } U_2]
\end{array}$$

It is not hard to show that no further simple rules apply, so the last relation

$$\mathsf{R} = \{[\top, x, \neg y, \neg U_1, U_2, \neg U_3], [\perp, \neg x, y, U_1, \neg U_2, U_3]\}$$

is 0-saturated. But we can say more that that. The relation consists of only the two determinate classes. Each variable of F_3 is in exactly one of the *TRUE*- and *FALSE*-classes, so R can be seen as a truth value assignment to the variables in $Vars(F_3)$. Furthermore, since R is 0-saturated these values are correctly propagated all the way up the formula to $U_3 = F_3$. This means that R defines a valuation on F_3, and this valuation places F_3 in the *FALSE*-class. In other words, F_3 is falsifiable and R gives a counter-model. $\diamond$

It is not always the case for a falsifiable formula F that dilemma yields a counter-model at zero level, but a falsifying valuation can always be found if we branch over all possible assignments to the variables of F. Conversely, if we ever reach a 0-saturated formula relation R with $|\mathsf{R}| = 2$ in a branch of a (possibly nested) dilemma rule application then we can terminate the attempt to prove F. Since $|\mathsf{R}| = 2$, all variables have been assigned values, and the fact that R is 0-saturated means that the relation defines a counter-model of F (the triplet representing F was placed in the *FALSE*-class at the outset of the derivation).

In our applications of the dilemma rule so far, we have branched only on the truth or falsehood of propositional variables or of the formula in question. We call such dilemma rule applications *atomic*. Proofs which employ only atomic applications of the dilemma rule are called *atomic dilemma proofs*. We now give an example of a 1-easy formula which becomes 2-hard if we allow only atomic dilemma rule applications.

Example 4.5 (1-depth non-atomic proof) The formula

$$F_4 := ((x \leftrightarrow y) \rightarrow (z \leftrightarrow w)) \leftrightarrow (\neg(z \leftrightarrow w) \rightarrow \neg(x \leftrightarrow y))$$

is a (moderately clever) way of disguising the tautological statement that an implication is equivalent to its contrapositive. Using the triplets

$$\begin{array}{lll} U_1 : x \leftrightarrow y & U_3 : U_1 \rightarrow U_2 & U_5 : U_3 \leftrightarrow U_4 \\ U_2 : z \leftrightarrow w & U_4 : \neg U_2 \rightarrow \neg U_1 & \end{array}$$

to represent F_4, we assume that U_5 is false and propagate. Because of the fact that the outermost connective is a bi-implication, the relation becomes 0-saturated after just one rule application:

$$\begin{array}{ll} \{[\top, \neg U_5], [x], [y], [z], [w], [U_1], [U_2], [U_3], [U_4]\} & [{F_4}^{\perp}] \\ \{[\top, \neg U_5], [x], [y], [z], [w], [U_1], [U_2], [U_3, \neg U_4]\} & [\text{rule (4.31b) on } U_5] \end{array}$$

If we branch on x, in the branch assuming x false we get the derivation

$$\begin{array}{ll} \{[\top, \neg x, \neg U_5], [y], [z], [w], [U_1], [U_2], [U_3, \neg U_4]\} & [x \equiv \perp] \\ \{[\top, \neg x, \neg U_5], [y, \neg U_1], [z], [w], [U_2], [U_3, \neg U_4]\} & [\text{rule (4.31f) on } U_1] \end{array}$$

and in the branch assuming x true we get:

$$\begin{array}{ll} \{[\top, x, \neg U_5], [y], [z], [w], [U_1], [U_2], [U_3, \neg U_4]\} & [x \equiv \top] \\ \{[\top, x, \neg U_5], [y, U_1], [z], [w], [U_2], [U_3, \neg U_4]\} & [\text{rule (4.31e) on } U_1] \end{array}$$

It is easy to see that the last formula relation in each branches is 0-saturated and that the formula relation intersection of the two relations is

$$\{[\top, \neg U_5], [x], [y], [z], [w], [U_1], [U_2], [U_3, \neg U_4]\},$$

which is the same relation as that preceding the dilemma rule application. That is, branching on x gives no new conclusions, and its easy to verify that the same holds for y, z and w. Thus, there is no 1-depth atomic dilemma proof of F_4.

If we instead branch on the equivalence of x and y we derive

$$\begin{array}{ll} \{[\top, \neg U_5], [x, y], [z], [w], [U_1], [U_2], [U_3, \neg U_4]\} & [x \equiv y] \\ \{[\top, U_1, \neg U_5], [x, y], [z], [w], [U_2], [U_3, \neg U_4]\} & [\text{rule (4.31g) on } U_1] \\ \{[\top, U_1, \neg U_5], [x, y], [z], [w], [U_2, \neg U_3, U_4]\} & [\text{rule (4.30e) on } U_4] \\ \{[\top, \mathbf{U_1}, \neg \mathbf{U_1}, \neg U_5], [x, y], [z], [w], [U_2, \neg U_3, U_4]\} & [\text{rule (4.30c) on } U_3] \\ \{[\top, \perp, x, \neg x, y, \neg y, \ldots, U_4, \neg U_4, U_5, \neg U_5]\} & [\perp_{F_4^+} \text{ by rule (4.18)}] \end{array}$$

in the branch assuming that x and y are equivalent and

$$\begin{array}{ll} \{[\top, \neg U_5], [x, \neg y], [z], [w], [U_1], [U_2], [U_3, \neg U_4]\} & [x \equiv \neg y] \\ \{[\top, \neg U_1, \neg U_5], [x, \neg y], [z], [w], [U_2], [U_3, \neg U_4]\} & [\text{rule (4.31h) on } U_1] \\ \{[\top, \neg U_1, U_3, \neg U_4, \neg U_5], [x, \neg y], [z], [w], [U_2]\} & [\text{rule (4.30d) on } U_3] \\ \{[\top, \mathbf{U_1}, \neg \mathbf{U_1}, U_3, \neg U_4, \neg U_5], [x, \neg y], [z], [w], [U_2]\} & [\text{rule (4.30b) on } U_4] \\ \{[\top, \perp, x, \neg x, y, \neg y, \ldots, U_4, \neg U_4, U_5, \neg U_5]\} & [\perp_{F_4^+} \text{ by rule (4.18)}] \end{array}$$

in the branch assuming that they are not. Both assumptions yield a contradiction, so $H(F_4) = 1$. ◇

In the last example we used the most general kind of dilemma rule assumptions, namely assumptions of type $P \equiv Q$ and $P \equiv Q^C$ for some subformulas P and Q of F. It is not hard to see that the proof above can be simplified by replacing the assumptions $x \equiv y$ and $x \equiv \neg y$ with assumptions about the truth or falsehood of $x \leftrightarrow y$, i.e. assumptions of type $P \equiv \bot$ and $P \equiv \top$. We will call the latter type of assumptions *bivalent*, since they make use of the bivalence principle (i.e. that P is either true or false). As we shall see presently, when Stålmarck's method searches for a proof or refutation of a formula it uses only bivalent dilemma rule applications.

We can formalize the reasoning in the last paragraph and define subsystems of dilemma with restrictions on which types of dilemma rule assumptions may occur. In order to be able to distinguish between the different kinds of dilemma derivations encountered in this section, we introduce terminology and notation for such restrictions of the dilemma proof system below.

Definition 4.28 (Atomic, bivalent and general dilemma) *Let* $\pi : \mathsf{R} \Rightarrow \mathsf{R}'$ *be a dilemma derivation as defined in definition 4.9.*

If the dilemma rule assumptions in π *are all of the form* $x \equiv \bot$ *or* $x \equiv \top$ *for* $x \in \mathit{Vars}(\mathsf{R})$*, we say that* π *is an* atomic *dilemma derivation. If all dilemma rule assumptions in* π *are of the form* $P \equiv \bot$ *or* $P \equiv \top$ *for* $P \in \mathit{Sub}(\mathsf{R})$*,* π *is said to be a* bivalent *derivation. Otherwise,* π *is a* general *(analytic) dilemma derivation.*

We let $\mathcal{D}$ *denote the general dilemma proof system (with the simple rules in figures A.2, A.3, A.4 and A.5 plus rules* (4.17) *and* (4.18) *and the dilemma rule* (4.19)*). The dilemma proof system restricted to atomic derivations only is denoted* $\mathcal{D}_A$ *and dilemma restricted to bivalent derivations is denoted* $\mathcal{D}_B$*.*

Also, we extend the definition of hardness in definition 4.15 in the natural way to atomic and bivalent dilemma, letting $H_{\mathcal{D}_A}(\mathsf{R})$ *denote the hardness of a formula relation* R *with respect to atomic dilemma* $\mathcal{D}_A$ *and* $H_{\mathcal{D}_B}(\mathsf{R})$ *the hardness with respect to bivalent dilemma* $\mathcal{D}_B$ *(and correspondingly for formulas* F*). When we need to make the distinction, we denote the hardness of a formula relation* R *in general dilemma* $H_{\mathcal{D}}(\mathsf{R})$*.*

Using this notation, for the formula F_4 in example 4.5 above it holds that $H_{\mathcal{D}_B}(F_4) = H_{\mathcal{D}}(F_4) = 1$, and it is easy to see that $H_{\mathcal{D}_A}(F_4) = 2$ (make nested atomic dilemma rule applications branching on x and y and then insert two copies each of the subderivations in the proof in the example).

The atomic and bivalent dilemma proof systems arise when we impose restrictions on which kind of assumptions may be branched on when introducing a dilemma rule application. Another way of modifying the proof system is to put restrictions on which conclusions may be drawn when *closing* a dilemma rule application. In dilemma we take the intersection of the consequences derived in the two branches. If we remove the "merging" part of the dilemma rule and limit rule applications to the special case where a contradiction is reached in one of the branches (schematically

$$\frac{\dfrac{\mathsf{R}}{\begin{array}{cc}\mathsf{R}[P \equiv Q] & \mathsf{R}[P \equiv Q^C] \\ \pi & \\ \bot & \end{array}}}{\mathsf{R}[P \equiv Q^C]} \tag{4.33}$$

and

$$\dfrac{\dfrac{\mathsf{R}}{\mathsf{R}[P \equiv Q] \quad \begin{matrix} \mathsf{R}[P \equiv Q^C] \\ \pi \\ \perp \end{matrix}}}{\mathsf{R}[P \equiv Q]} \tag{4.34}$$

for contradictions in the left and right branches respectively), we see that this is equivalent to the *reductio ad absurdum* or *RAA rule*. Consequently, the proof system using this branching rule is called the *RAA proof system*.

Definition 4.29 (RAA proof systems) *Let* $\pi : \mathsf{R} \Rightarrow \mathsf{R}'$ *be a dilemma derivation according to definition 4.9.*

If all dilemma rule applications in π *derive a contradiction in at least one of the two branches, i.e. are all of the form* (4.33) *or* (4.34)*, we say that* π *is an* RAA derivation.

The dilemma proof system restricted to RAA derivations only is called the reductio *or* RAA proof system *and is denoted* $\mathcal{RAA}$*. The hardness of a formula relation* R *(or formula* F*) with respect to the RAA proof system is denoted* $H_{\mathcal{RAA}}(\mathsf{R})$ *(or* $H_{\mathcal{RAA}}(F)$*). The atomic and bivalent varieties of* $\mathcal{RAA}$ *(defined in analogy with definition 4.28) are denoted* $\mathcal{RAA}_A$ *and* $\mathcal{RAA}_B$*, respectively.*

4.2.2 κ-saturation and Proof Methods

The proof systems in definitions 4.28 and 4.29 can be compared with respect to the minimal depth, length or size of proofs of formulas. The object of such analysis is to prove that one system is at least as strong as another with respect to proof depth, length or size or find families of formulas F_n that separate two systems with respect to one or more of these parameters. We will study these questions in more detail in chapter 6. At the moment, our aim is to present a general framework for constructing proof search algorithms in these systems and to analyze the two proof search algorithms which are most interesting from a practical point of view.

We first give the intuition behind the construction of our proof methods. As in definition 4.28 above, let $\mathcal{D}$ denote the full dilemma proof system. The idea is to construct a proof method by defining a sequence of increasingly powerful subsystems of $\mathcal{D}$ and devising proof search algorithms for these subsystems.

Let $\mathcal{D}_0$ be $\mathcal{D}$ without the dilemma rule and let $\mathcal{D}_{i+1}$ be $\mathcal{D}$ with the restriction that derivations in applications of the dilemma rule should be $\mathcal{D}_i$-derivations. That is, proofs in $\mathcal{D}_1$ have at most one open assumption, proofs in $\mathcal{D}_2$ at most two open assumptions etc. Then Stålmarck's method can be seen as a family of algorithms which search exhaustively for proofs in $\mathcal{D}_i$ for increasing i.

Proofs in $\mathcal{D}_0$ can be found in linear time by taking the closure of the simple rules. The time required to search exhaustively for a proof in $\mathcal{D}_i$ is $\mathrm{O}\left(s^{\mathrm{O}(i)}\right)$, where s is the size of the formula. Empirically, many industrial verification problems give rise to formulas with proofs in $\mathcal{D}_0$ or $\mathcal{D}_1$. Such formulas may be large, but this need not be a problem since the proof method copes well with even very large formulas as long as the hardness degree is low.

The algorithm that searches exhaustively for proofs in $\mathcal{D}_\kappa$ for $\kappa \geq 0$ is called *κ-saturation*. In order to present the algorithm more formally we need a couple of definitions.

Definition 4.30 (κ-consequence) *Let $\mathcal{P}$ denote one of the dilemma proof systems $\mathcal{D}$, $\mathcal{D}_B$ or $\mathcal{D}_A$ or one of the reductio proof systems $\mathcal{RAA}$, $\mathcal{RAA}_B$ or $\mathcal{RAA}_A$ and let R and R' be compatible formula relations. We say that R' is a* κ-consequence *of R (with respect to the proof system $\mathcal{P}$) if there is a derivation $\pi : \mathsf{R} \Rightarrow \mathsf{R}'$ in $\mathcal{P}$ with $D(\pi) \leq \kappa$.*

If the relations $\mathsf{R}[\Psi]$ and $\mathsf{R}[\Phi]$ are κ-consequences of R, then so is $\mathsf{R}[\Psi, \Phi]$. For if $\pi_1 : \mathsf{R} \Rightarrow \mathsf{R}[\Psi]$ and $\pi_2 : \mathsf{R} \Rightarrow \mathsf{R}[\Phi]$ are derivations of depth at most κ, then the composition $\pi_1 \bullet (\pi_2[\Psi]) : \mathsf{R} \Rightarrow \mathsf{R}[\Psi, \Phi]$ is a (possibly non-proper and non-standardized) derivation of $\mathsf{R}[\Psi, \Phi]$ of depth $\max\{D(\pi_1), D(\pi_2)\} \leq \kappa$. If we order the set of κ-consequences of R with respect to $\sqsubseteq$, it follows (since the set of formula relations on $Sub(\mathsf{R})$ is finite) that there is a unique maximal κ-consequence of R. This relation is called the *κ-saturation* of R.

Definition 4.31 (κ-saturation) *Let $\mathcal{P}$ denote one of $\mathcal{D}$, $\mathcal{D}_B$ or $\mathcal{D}_A$ or one of $\mathcal{RAA}$, $\mathcal{RAA}_B$ or $\mathcal{RAA}_A$.*

The κ-saturation *of a relation R with respect to $\mathcal{P}$, denoted $Sat_{\mathcal{P}}(\mathsf{R}, \kappa)$ or just $Sat(\mathsf{R}, \kappa)$ when the meaning is clear from context, is the closure of R under κ-consequence in $\mathcal{P}$. That is, $Sat_{\mathcal{P}}(\mathsf{R}, \kappa)$ is a κ-consequence of R, and for all κ-consequences R' of R it holds that $\mathsf{R}' \sqsubseteq Sat_{\mathcal{P}}(\mathsf{R}, \kappa)$.*

R is said to be κ-saturated *with respect to $\mathcal{P}$ if $\mathsf{R} = Sat_{\mathcal{P}}(\mathsf{R}, \kappa)$.*

In other words, a relation R is κ-saturated if for any derivation $\pi : \mathsf{R} \Rightarrow \mathsf{R}'$ of depth $D(\pi) \leq \kappa$ it holds that $\mathsf{R}' = \mathsf{R}$. If a relation R has hardness degree $H(\mathsf{R}) \leq \kappa$, it follows that $Sat(\mathsf{R}, \kappa)$ must be explicitly contradictory.

The central routine in Stålmarck's method is a procedure $\texttt{saturate}(\mathsf{R}, \kappa)$ which κ-saturates relations R with respect to (bivalent) dilemma and returns the κ-saturated relation. In view of the last paragraph, this can be seen to be equivalent to searching exhaustively for a κ-depth bivalent dilemma proof.

In our implementation of this procedure, we use one unique representative for each equivalence class (which is vital for the time complexity of the procedure, as we shall see later when we analyze it). One way of looking at this is that we let the algorithm operate on *equivalence classes* instead of triplets. Before presenting the saturation algorithm, we introduce some notation for this.

We use $Compound_{Cl}(\mathsf{R})$ to denote the set of *compound equivalence classes* of R, i.e. the equivalence classes containing compound subformulas. More formally, $Compound_{Cl}(\mathsf{R})$ is defined by

$$Compound_{Cl}(\mathsf{R}) := \{\mathsf{C} \in \mathsf{R} \mid \exists P \in Compound(\mathsf{R}) \text{ such that } P \in \mathsf{C}\}. \quad (4.35)$$

The set of *atomic equivalence classes* of R, denoted $Var_{Cl}(\mathsf{R})$, is defined by

$$Var_{Cl}(\mathsf{R}) := \{\mathsf{C} \in \mathsf{R} \mid \exists x \in Vars(\mathsf{R}) \text{ such that } x \in \mathsf{C}\} \quad (4.36)$$

(note that according to these definitions, an equivalence class can be both atomic and compound).

For equivalence classes $\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3$ of a relation R and a fix $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, we say that the ternary relation $\langle \mathsf{C}_1 : \mathsf{C}_2 \circ \mathsf{C}_3 \rangle$ holds if for $i = 1, 2, 3$ there exist $U_i \in \mathsf{C}_i$ such that $U_1 : U_2 \circ U_3$ is a triplet in R. If $\langle \mathsf{C}_1 : \mathsf{C}_2 \circ \mathsf{C}_3 \rangle$ holds, we call C_1 a *parent* of C_2 and C_3. The equivalence classes C_2 and C_3 are called the (left and right) *children* of C_1.

$\texttt{saturate}(\mathsf{R}, 0) =$
$\quad S := \mathit{Compound}_{Cl}(\mathsf{R})$
$\quad$ while $S \neq \emptyset$ and not $(\texttt{contradictory}(\mathsf{R}))$
$\quad\quad \mathsf{C} := \texttt{retrieve}(S)$
$\quad\quad (\mathsf{R}, N) := \texttt{investigate}(\mathsf{R}, \mathsf{C})$
$\quad\quad S := S \cup N$
$\quad \texttt{return}(\mathsf{R})$

Figure 4.5: Pseudocode for 0-saturation.

$\texttt{investigate}(\mathsf{R}, \mathsf{C}) =$
$\quad N := \emptyset$
$\quad$ For all $\mathsf{D}_1, \mathsf{D}_2 \in \mathsf{R}$ and $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ such that $\langle \mathsf{C} : \mathsf{D}_1 \circ \mathsf{D}_2 \rangle$
$\quad\quad \mathsf{R}' := \texttt{propagate}(\mathsf{R}, \mathsf{C}, \mathsf{D}_1, \circ, \mathsf{D}_2)$
$\quad\quad$ if $\mathsf{R}' \neq \mathsf{R}$
$\quad\quad\quad N := N \cup \texttt{affected}(\mathsf{R}', \mathsf{R})$
$\quad\quad\quad \mathsf{R} := \mathsf{R}'$
$\quad \texttt{return}(\mathsf{R}, N)$

Figure 4.6: Pseudocode for 0-saturation subroutine.

For an equivalence class C of a formula relation R, the set of parent classes of C, denoted $\mathit{Parents}_{Cl}(\mathsf{R})$, is defined to be all $\mathsf{D} \in \mathsf{R}$ for which there exist $\mathsf{C}' \in \mathsf{R}$ and $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ such that either $\langle \mathsf{D} : \mathsf{C} \circ \mathsf{C}' \rangle$ or $\langle \mathsf{D} : \mathsf{C}' \circ \mathsf{C} \rangle$ holds.

We extend these definitions to triplets of type $U_1 : U_2 \circ \neg U_3$, $U_1 : \neg U_2 \circ U_3$ and $U_1 : \neg U_2 \circ \neg U_3$ in the natural way.

0-saturation

The idea behind 0-saturation is to apply propagation rules to a formula relation until no further new associations can be derived. We start by inserting all compound triplet variables (i.e. triplets which represent compound subformulas) into a pool, and then look at all the triplets in the pool in turn and try to apply simple rules. After a successful rule application, all triplets affected by the application of the rule are reinserted into the pool so that it can be investigated whether any further associations can now be derived from the enlarged formula relation. This continues until the pool of triplets is empty, at which point the resulting 0-saturated formula relation is returned. A pseudocode description of 0-saturation (operating on equivalence classes instead of triplets) is given in figures 4.5 and 4.6. For simplicity, we pass function arguments by value.

In figure 4.5, the function $\texttt{contradictory}(\mathsf{R})$ returns *TRUE* if R is the canonical contradictory relation with $|\mathsf{R}| = 1$. $\texttt{retrieve}(S)$ returns the next equivalence class in the pool S (according to some ordering) and deletes it from the pool. $\texttt{investigate}(\mathsf{R}, \mathsf{C})$ tries to derive new associations on R from the triplets in the equivalence class C. If `investigate` is successful, it returns the enlarged formula relation and a set N of equivalence classes that should be inserted into the pool S for further investigations.

Figure 4.6 describes the `investigate` subroutine in greater detail. For each relation $\langle \mathsf{C} : \mathsf{D}_1 \circ \mathsf{D}_2 \rangle$, the procedure calls $\texttt{propagate}(\mathsf{R}, \mathsf{C}, \mathsf{D}_1, \circ, \mathsf{D}_2)$, which investigates whether any propagation rules are applicable to this relation, and if so applies them. If a new association was derived, `investigate` then determines which equivalence classes should be added back to the pool.

But how do we determine which equivalence classes should be put back into the pool? Suppose that some propagation rule is successfully applied and associates two distinct equivalence classes C_1 and C_2 of R. Let R' denote the new formula relation containing $\mathsf{C}_1 \cup \mathsf{C}_2$. Now we want to insert into the pool S all equivalence classes that might be affected by the association of C_1 and C_2, i.e. all equivalence classes D for which the equivalence of C_1 and C_2 might lead to the derivation of further equivalences involving D.

A first suggestion is to add to the pool S the new equivalence class $\mathsf{C}_1 \cup \mathsf{C}_2$ (to let the new information propagate downwards in the formula to children of C_1 and C_2) and all parents of C_1 and C_2 (to let the information propagate upwards). In this way we cover all equivalence classes that can possibly be affected by the newly derived equivalence.

We can be smarter than that, however. To see how, consider first a triplet $U_1 : U_2 \circ U_3$ in R. The question is in what cases the merging of the equivalence classes C_1 and C_2 can affect the triplet U_1 so that a propagation rule which was not applicable before can now be applied on U_1.

As we mentioned in section 4.2.1 when giving examples of dilemma derivations, a propagation rule can be applicable to a triplet U_1 only if either one of the U_i is equivalent to $\top$ or $\bot$ (i.e. is in a determinate equivalence class) or $U_i \equiv U_j$ (or $U_i \equiv {U_j}^C$) for $i \neq j$. Consequently, only triplets where one of the U_i is placed in a determinate equivalence class or where two U_i, U_j are placed in the same indeterminate equivalence class (or complementary indeterminate equivalence classes) are candidates for new propagation rule applications.

If we apply the above reasoning to our equivalence classes C_1 and C_2, we see that:

- if $\top \in \mathsf{C}_1$ or $\bot \in \mathsf{C}_1$, then $\mathsf{C}_1 \cup \mathsf{C}_2$ and all equivalence classes in the set $\mathit{Parents}_{Cl}\,(\mathsf{C}_2)$ should be placed in the pool (and vice versa if $\top \in \mathsf{C}_2$ or $\bot \in \mathsf{C}_2$),
- otherwise, $\mathsf{C}_1 \cup \mathsf{C}_2$ should be added if C_1 is a parent of C_2 or ${\mathsf{C}_2}^C$ (or vice versa), as should all $\mathsf{D} \in \mathit{Parents}_{Cl}\,(\mathsf{C}_1) \cap \mathit{Parents}_{Cl}\,(\mathsf{C}_2)$ such that one of $\langle \mathsf{D} : \mathsf{C}_1 \circ \mathsf{C}_2 \rangle$, $\langle \mathsf{D} : {\mathsf{C}_1}^C \circ \mathsf{C}_2 \rangle$, $\langle \mathsf{D} : \mathsf{C}_1 \circ {\mathsf{C}_2}^C \rangle$ or $\langle \mathsf{D} : {\mathsf{C}_1}^C \circ {\mathsf{C}_2}^C \rangle$ holds (or vice versa with C_1 and C_2 interchanged).

The function which takes R' and R and computes which equivalence classes are affected by the new associations derived and should be inserted in the pool is called $\texttt{affected}(\mathsf{R}', \mathsf{R})$.

After having merged two equivalence classes C_1 and C_2, we replace all references to the two classes with references to $\mathsf{C}_1 \cup \mathsf{C}_2$. In this way we make sure that each equivalence class is represented by one unique representative.

When the pool is empty (or a contradiction has been derived) in figure 4.5 it must be the case that no further associations can be derived by propagation. Thus, the formula relation is 0-saturated and the algorithm terminates. (Note that the concept of 0-saturation is independent of the dilemma or reductio proof system used, since the propagation rules are the same for all the systems).

We now analyze the time complexity of the 0-saturation algorithm. Clearly, the critical part is the innermost loop in `investigate`(R, C) in figure 4.6. Each relation $\langle \mathsf{C} : \mathsf{D}_1 \circ \mathsf{D}_2 \rangle$ in an equivalence class C corresponds to some triplet $U_1 : U_2 \circ U_3$. If we keep a list of the triplets in C, we can implement the for-loop in `investigate` efficiently by iterating over this list. As a further optimization, we can add a flag for each triplet which is set to *FALSE* when we know that the relations between the triplet variables are such that the triplet cannot yield any new information. For such triplets no call to `propagate` is made, and we consider the cost of checking the flags to be negligible in the time complexity calculations.

For the triplets $U_1 : U_2 \circ U_3$ that generate a call to `propagate`, the relation $\langle \mathsf{C} : \mathsf{D}_1 \circ \mathsf{D}_2 \rangle$ corresponding to the triplet is investigated to see if any simple rules apply. We say that the triplet is *evaluated*. It is not hard to see that triplets can be evaluated in constant time. If we accept that the calls to `affected` ensuing on a successful evaluation can be completed in constant time (which of course is not trivial and should be elaborated on, but we skip this and refer to [49] for more low-level implementation details) it follows that the cost of a call to `investigate` can be measured by the number of calls from within this procedure to `propagate`.

The total cost of 0-saturation can thus be measured by the total number of calls to `propagate`. We claim that this number is linear in the number of triplets, i.e. in $|Sub\,(\mathsf{R})|$.

To see why, consider how many times a triplet $U_1 : U_2 \circ U_3$ can be evaluated by `propagate`. We say that a triplet $U_1 : U_2 \circ U_3$ is *triggered* in the course of a saturation when any two elements in $\{U_1, U_2, U_3, \neg U_1, \neg U_2, \neg U_3, \top, \bot\}$ become equivalent. The equivalence class containing a triggered triplet is placed in the pool S to be evaluated using the propagation rules. After an evaluation, a triplet need not be evaluated again until the relations between the triplet variables have changed, and a triplet that can no longer propagate any information can be discarded for the rest of the duration of the saturation. If we augment our triplet flag with a note specifying the reason why the triplet is not interesting and avoid evaluating it again until this has changed, we can get an upper limit on the number of times a triplet is evaluated during 0-saturation.

We divide our analysis into different cases depending on how many of the U_i have been assigned truth values (i.e. placed in determinate equivalence classes) and how many distinct non-complementary equivalence classes the remaining U_j represent:

- If U_1, U_2 and U_3 all belong to distinct indeterminate equivalence classes, there is no point in evaluating the triplet $U_1 : U_2 \circ U_3$. No calls to `propagate` will be made for such triplets U_1.

- If the U_i belong to two distinct non-complementary equivalence classes, we evaluate U_1. If a new association is derived, the number of distinct non-complementary indeterminate equivalence classes represented by the U_i decreases by one. If not, we mark the triplet with *FALSE* and make a note not to evaluate it again until the number of distinct non-complementary indeterminate equivalence classes has been decreased by one.

- If one of the U_i has been assigned, we try to propagate to the other variables in U_1. Again, if the evaluation is successful, the number of distinct non-complementary indeterminate equivalence classes decreases by one, and otherwise we make a note not to try again with this triplet until this happens (as a result of some other propagation).

- If two of the triplet variables U_i have been assigned, the triplet either propagates a truth value to the third variable or is useless. We evaluate it to determine which of the cases hold and then mark it *FALSE* and throw it away for good.
- If U_1, U_2 and U_3 all belong to just one equivalence class and its complement, the triplet either propagates a truth value to all of the variables or is useless. Again, we evaluate the triplet and then mark it *FALSE* and throw it away.

From this case analysis it follows that the number of evaluations in `propagate` caused by each triplet can be bounded by a constant (in fact, the maximum number of evaluations is 2). Thus, the total number of calls to `propagate` during 0-saturation is linear in the number of triplets in R.

By applying some further clever optimizations not discussed above, one can implement the algorithm in figures 4.5 and 4.6 to run in linear time [49, 51]. We do not give any details, but state the fact as a lemma for reference.

Lemma 4.32 (Time complexity of 0-saturation)
0-saturation of a formula relation R *can be performed in time* $\mathrm{O}\left(|\mathit{Sub}\left(\mathsf{R}\right)|\right)$.

(κ+1)-saturation

(κ+1)-saturation is defined inductively in terms of κ-saturation and branching. Because of the dependence on branching, we get different (κ+1)-saturation algorithms depending on which rules for branching and merging of the branches we use.

The general idea behind (κ+1)-saturation is simple: In one iteration, we branch over all pairs of admissible complementary assumptions (atomic, bivalent or general) in turn. For each pair of assumptions, we κ-saturate the two branches and then merge them according to the dilemma or reductio proof system rules. If the formula relation is enlarged during such an iteration, we repeat the whole iteration once more. If an iteration results in no new consequences, the formula relation must be (κ+1)-saturated with respect to the proof system used and the algorithm terminates.

Although the (κ+1)-saturation algorithm is guaranteed to find as shallow a proof as possible in all proof systems discussed in this chapter, the variants for reductio are more of a theoretical interest. If we have taken the time to κ-saturate the formula relations in the two branches, the extra cost of computing their intersection is insignificant, so there is not much point in designing “undirected” exhaustive search reductio κ-saturation algorithms. See section 4.2.3 for a brief discussion of a directed proof search algorithm for reductio.

In this section we present and analyze the two variants of (κ+1)-saturation that are most interesting from an applied perspective: *general dilemma (κ+1)-saturation*, which branches on the equivalence of all pairs of equivalence classes in a formula relation (pseudocode given in figure 4.7), and *bivalent dilemma (κ+1)-saturation*, which branches on the truth or falsehood of each

$$
\begin{array}{l}
\texttt{saturate}_{\mathcal{D}}\big(\mathsf{R}, \kappa+1\big) \;= \\
\quad\quad \text{repeat} \\
\quad\quad\quad\quad \mathsf{R}' := \mathsf{R} \\
\quad\quad\quad\quad i := 1 \\
\quad\quad\quad\quad \text{while } i < |\mathsf{R}|/2 \\
\quad\quad\quad\quad\quad\quad j := i+1 \\
\quad\quad\quad\quad\quad\quad \text{while } j \leq |\mathsf{R}|/2 \\
\quad\quad\quad\quad\quad\quad\quad\quad \mathsf{R}_1 := \texttt{saturate}_{\mathcal{D}}\big(\mathsf{R}\big[\mathsf{C}_i \equiv \mathsf{C}_j\big], \kappa\big) \\
\quad\quad\quad\quad\quad\quad\quad\quad \mathsf{R}_2 := \texttt{saturate}_{\mathcal{D}}\big(\mathsf{R}\big[\mathsf{C}_i \equiv \mathsf{C}_j{}^{C}\big], \kappa\big) \\
\quad\quad\quad\quad\quad\quad\quad\quad \mathsf{R} := \mathsf{R}_1 \sqcap \mathsf{R}_2 \\
\quad\quad\quad\quad\quad\quad\quad\quad j := j+1 \\
\quad\quad\quad\quad\quad\quad i := i+1 \\
\quad\quad \text{until } \mathsf{R}' = \mathsf{R} \text{ or } \texttt{contradictory}\big(\mathsf{R}\big) \\
\quad\quad \texttt{return}\big(\mathsf{R}\big)
\end{array}
$$

Figure 4.7: Pseudocode for general dilemma (κ+1)-saturation.

$$
\begin{array}{l}
\texttt{saturate}_{\mathcal{D}_B}\big(\mathsf{R}, \kappa+1\big) \;= \\
\quad\quad \text{repeat} \\
\quad\quad\quad\quad \mathsf{R}' := \mathsf{R} \\
\quad\quad\quad\quad i := 2 \\
\quad\quad\quad\quad \text{while } i \leq |\mathsf{R}|/2 \\
\quad\quad\quad\quad\quad\quad \mathsf{R}_1 := \texttt{saturate}_{\mathcal{D}_B}\big(\mathsf{R}\big[\mathsf{C}_i \equiv \mathsf{C}_\top\big], \kappa\big) \\
\quad\quad\quad\quad\quad\quad \mathsf{R}_2 := \texttt{saturate}_{\mathcal{D}_B}\big(\mathsf{R}\big[\mathsf{C}_i \equiv \mathsf{C}_\bot\big], \kappa\big) \\
\quad\quad\quad\quad\quad\quad \mathsf{R} := \mathsf{R}_1 \sqcap \mathsf{R}_2 \\
\quad\quad\quad\quad\quad\quad i := i+1 \\
\quad\quad \text{until } \mathsf{R}' = \mathsf{R} \text{ or } \texttt{contradictory}\big(\mathsf{R}\big) \\
\quad\quad \texttt{return}\big(\mathsf{R}\big)
\end{array}
$$

Figure 4.8: Pseudocode for bivalent dilemma (κ+1)-saturation.

equivalence class (pseudocode in figure 4.8). Note that if we want to implement atomic (κ+1)-saturation, we can change the pseudocode in figure 4.8 so that it loops over equivalence classes in $Var_{Cl}(\mathsf{R})$ only.

In figures 4.7 and 4.8, we consider the formula relations to be ordered sets of equivalence classes $\mathsf{R} = \{\mathsf{C}_1, \ldots, \mathsf{C}_n, \mathsf{C}_{-1}, \ldots, \mathsf{C}_{-n},\}$ with $|\mathsf{R}| = 2n$, where $\mathsf{C}_1 = \mathsf{C}_\top$ is the *TRUE*-class, $\mathsf{C}_{-1} = \mathsf{C}_\bot$ is the *FALSE*-class and more generally ${\mathsf{C}_i}^C = \mathsf{C}_{-i}$. When two equivalence classes C_i and C_j (where we assume $|i| < |j|$) are merged, we set $\mathsf{C}_i := \mathsf{C}_i \cup \mathsf{C}_j$ and renumber the other equivalence classes as needed.

As to the complexity of these algorithms, note that each iteration in the saturation algorithms decreases the number of equivalence classes by at least two (when this is no longer true the algorithms terminate), so there will be a total of at most $|\mathsf{R}|/2$ iterations.

In general (κ+1)-saturation, each iteration performs at most $\left(|\mathsf{R}|/2\right)^2$ passes of κ-saturation, and in bivalent (κ+1)-saturation the maximum number of κ-saturations is $|\mathsf{R}|/2$. Observing that $|\mathsf{R}| = \mathrm{O}\left(|Sub\,(\mathsf{R})|\right)$, by lemma 4.32 and induction we get:

Theorem 4.33 (Time complexity of κ-saturation)
General dilemma κ-saturation of a formula relation R *can be performed in time* $\mathrm{O}\left(|Sub\,(\mathsf{R})|^{3\kappa+1}\right)$.

Bivalent dilemma κ-saturation of a formula relation R *can be performed in time* $\mathrm{O}\left(|Sub\,(\mathsf{R})|^{2\kappa+1}\right)$.

See [49] for a more detailed upper bound on bivalent dilemma κ-saturation.

The Proof Method

Given a supposedly tautological (or contradictory) formula F, how do we find a proof (or refutation) of F efficiently?

In view of what has been said earlier in this chapter, it seems to be a good idea to minimize the branching in the proof and try to find as shallow a proof as possible. So we start with 0-saturation. If 0-saturation give no conclusive results, we try 1-saturation, then 2-saturation and so on.

The Stålmarck proof method κ-saturates the relation for the formula in the dilemma proof system for increasing κ until it finds either a proof or a counter-model. Of course, information gathered during κ-saturation is available during subsequent (κ+1)-saturation. We call the variants of Stålmarck's method using atomic, bivalent or general dilemma the atomic, bivalent or general Stålmarck's method, respectively.

The cost of finding a proof or refutation for a exactly κ-hard formula F is essentially that of performing κ-saturation of the corresponding formula relation.

Theorem 4.34 (Time complexity of Stålmarck's method)
Let F be a tautological formula in propositional logic.

Then the general Stålmarck's method finds a proof of F in time

$$\mathrm{O}\left(S\,(F)^{3\cdot H_{\mathcal{D}}(F)+1}\right).$$

The bivalent Stålmarck's method finds a proof in time

$$\mathrm{O}\left(S\,(F)^{2\cdot H_{\mathcal{D}_B}(F)+1}\right),$$

which in terms of the hardness $H_{\mathcal{D}}$ in general dilemma cannot be worse than $\mathrm{O}\big(S(F)^{4\cdot H_{\mathcal{D}}(F)+1}\big)$.

Proof: Let $\kappa := H_{\mathcal{D}}(F)$ and $\mathsf{R} := F^{\perp}$.

Suppose first that no consequences whatsoever are derived until κ-saturation starts. In this case we will have had one failed iteration each of 0-saturation, 1-saturation, ..., (κ-1)-saturation. By (the reasoning preceding) theorem 4.33, the cost of all these failed iterations added together must be less than that of one iteration in κ-saturation. Consequently, the total cost is at most $|\mathsf{R}|/2+1$ iterations of κ-saturation.

If instead κ'-saturation is successful for some $\kappa' < \kappa$, then each such (full) κ'-saturation is cheaper than an iteration of κ-saturation and decreases $|\mathsf{R}|$ by at least two (and thus the number of possible remaining κ-saturations by at least one). We see that a successful κ'-saturation can only lower the upper bound given in the first case above, which is the worst case.

The theorem follows if we apply theorem 4.33 on our derived upper bound and note that $\left|F^{\perp}\right| \leq 2 \cdot S(F)$ and $H_{\mathcal{D}_B}(F) \leq 2 \cdot H_{\mathcal{D}}(F)$. □

Although general saturation has better asymptotic properties than bivalent saturation in theory, the latter seems to achieve superior performance in practice. Therefore bivalent dilemma saturation is the version of the saturation algorithm used in commercial implementations of Stålmarck's method [51].

If we compare corollary 4.25 and theorem 4.34, we see that the time complexity of Stålmarck's method is less than cubic in the upper bound on proof length (less than quartic for the bivalent variant). To show that Stålmarck's method is an efficient proof search algorithm, however, we would like to bound the time complexity as a function of a *lower* bound on proofs. In the next theorem, we prove that the algorithm finds proofs in time quasi-polynomial in the size of a smallest proof. That is, dilemma is a quasi-automatizable proof system (definition 2.18 on page 15).

Theorem 4.35
General and bivalent dilemma are both quasi-automatizable proof systems.

Proof: Let F be a tautology in propositional logic.

Using corollaries 4.21 and 4.25 and theorem 4.34, the same technique as in the proof of theorem 4.26 can be used to show that the *bivalent* Stålmarck's method finds a (bivalent) proof of F in time quasi-polynomial in the size of a smallest *general* dilemma proof. Thus general dilemma is quasi-automatizable.

Since a smallest bivalent dilemma proof certainly is at least as large as a smallest general proof, this shows that bivalent dilemma is quasi-automatizable as well. □

Remark 4.36 For readers not willing to accept lemma 4.32 without a full proof, we note that there is an easier way to establish theorem 4.35. To make the proof above go through, it is sufficient to show that κ-saturation can be performed in time $\mathrm{O}\left(S(F)^{c_1 \cdot H(F)+c_2}\right)$ for some constants c_1 and c_2. Perhaps the easiest way of proving this bound is to show that 0-saturation can be performed not necessarily in linear but in polynomial time (which is trivial), replace the reference to lemma 4.32 in the proof of theorem 4.33 by this polynomial bound and then reuse the rest of the proofs of theorems 4.33 and 4.34.

4.2.3 Extensions of Stålmarck's Method

Stålmarck's method has been implemented in the tool *Prover Plug-in*™, which in turn is the proof search routine (or *proof engine*) used in a number of products from Prover Technology AB. Below follows a short overview of some the types of optimizations applied in Prover Plug-in™. For a more detailed discussion on this subject we refer to [44].

Formula Transformation

The Stålmarck method as defined in this chapter operates on formulas containing all the usual logical connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. When implementing the method, to simplify the algorithm the formulas are transformed to equivalent formulas on fewer connectives. The exact method chosen for rewriting the formulas is not of crucial importance, but it should avoid the blow-up in formula size that results from splitting logical equivalences.

An early implementation of Stålmarck's algorithm systematically pulled negations up the formula, leaving a body involving only conjunction, disjunction, implication and bi-implication applied to unnegated propositional variables [55]. One advantage of rewriting the formula in this way is that if the resulting formula is negated, we know immediately that it cannot be a tautology (set all variables to *TRUE*). For a description of this transformation (and a further reduction to a formula over just $\{\neg, \rightarrow\}$), see [52]. A rather detailed description of another implementation which uses a transformation to formulas over $\{\neg, \wedge, \leftrightarrow\}$ is given in [32]. In the current version of Prover Plug-in™, the internal representation of formulas are over the set of connectives $\{\neg, \rightarrow, \leftrightarrow\}$ [51].

Representation of Formula Relations

As has already been noted in sections 4.2.1 and 4.2.2, since formula relations respect complement, it is sufficient to represent a formula relation by only half of its equivalence classes. Also, in implementations each equivalence class in a formula relation is represented by a *unique* representative, which is important for the complexity of the proof search algorithm.

Composite rules

In our definition of the dilemma proof system, all propagation rules are simple, i.e. they act on the triplet representing a subformula $P \circ Q$ and its immediate subformulas P and Q. But one can also devise rules that apply not on just a single subformula but on *pairs* of subformulas (while still not requiring any branching). Such rules are called *composite* propagation rules.

For example, suppose that it has been derived that two subformulas $F \rightarrow G$ and $G \rightarrow F$ have the same truth value. Then it is easy to verify that the subformulas F and G must have the same truth value as well (and consequently both implications are true). If we generalize this rule somewhat and write it down in the notation used for the simple rules, we have:

$$\frac{P \equiv S \quad Q \equiv R \quad P \rightarrow Q \equiv R \rightarrow S}{P \equiv Q} \tag{4.37}$$

In dilemma as it is described in section 4.1, we would need to insert a 1-depth derivation branching over P or Q to deduce this equivalence.

Another example of a composite rule is the rule stating that an implication $F \to G$ is equivalent to its contrapositive $\neg G \to \neg F$:

$$\frac{P \equiv S^C \quad Q \equiv R^C}{P \to Q \equiv R \to S} \tag{4.38}$$

If we augment the dilemma proof system with this rule, the 1-hard formula in example 4.5 becomes 0-easy.

As these two examples show, composite rules can be used to enhance proof power by reducing the hardness of a formula. Rules such as (4.37) and (4.38) decrease the hardness degree with (at most) a constant, which means that they are of little interest from a theoretical point of view. In practice, however, they can have a large impact. As we have seen in section 4.2.2, Stålmarck's method is sensitive to the hardness degree. Adding composite rules to the proof system increases the complexity of the saturation algorithm, but if we can avoid $(\kappa{+}1)$-saturation by proving a formula in depth κ with the use of composite rules, it might be worth the extra cost. (Of course, to be able to use this type of rules we need some kind of data structure, for instance a hash table, where we can search for "companion triplets", i.e. triplets such as the ones representing $F \to G$ and $G \to F$ in the first example above).

Also, composite rules can be used during formula parsing and construction of the triplet set to share triplets for semantically equivalent subformulas. If we run across the two subformulas $F \to G$ and $\neg G \to \neg F$, we know by rule (4.38) that they are equivalent. Therefore, we do not have to build separate triplets for these two formulas but can let them share the same triplet (which can be seen as inserting a dilemma derivation step placing the subformulas in the same equivalence class "on the fly").

Implications and Implication Graphs

The formula relations used in dilemma contain information about equivalences between subformulas. In Prover Plug-in™it is possible to keep record not only of equivalences but also of *formula implications* $P \Rightarrow Q$ between subformulas. The interpretation of this notation is that if we derive that P is true, then Q must be true also. If $P \Rightarrow Q$ and $Q \Rightarrow P$ then $P \equiv Q$.

The rules for deriving formula implications are of three kinds:

Simple implication rules One example of a simple rule is $P \wedge Q \Rightarrow P$ (such rules can be applied when parsing a formula and building the triplets).

Transitivity rule If $P \Rightarrow Q$ and $Q \Rightarrow R$ then $P \Rightarrow R$.

Introduction rule If we derive $Q \equiv \top$ under the assumption $P \equiv \top$ then it holds that $P \Rightarrow Q$.

The main motivation for formula implications is that on the basis of them one can construct implication graphs which can be used to find efficient variable orderings for backtracking (see below).

Backtracking

In industrial applications, if 1-saturation of a formula relation $F^{\perp}$ does not yield a contradiction, there might be reason to suspect that the formula F is falsifiable. In this case, Prover Plug-in™can be instructed to search for a counter-model (i.e. a satisfying valuation of $\neg F$) by *backtracking*. One can also switch to backtracking with the purpose of searching for a proof if one still believes that F is valid but it is simply too expensive to continue the proof search by 2-saturating the formula relation.

Backtracking works as follows: Based on some ordering of the variables (chosen for instance by considering the number of occurrences or information from an implication graph), the backtracking procedure divides the problem into two subproblems in which the first variable with respect to the chosen order is assumed false and true respectively. The assumption is propagated by 0-saturation, and if no contradiction or counter-model is derived the procedure then branches recursively on the truth value of the second variable, and so on.

If in any branch we derive a 0-saturated formula relation R with $|\mathsf{R}| = 2$, that formula relation gives a counter-model for F. When we reach an explicitly contradictory relation in a branch, that branch is terminated. If as a result contradictions are derived in all branches, the formula F must be valid and the derivations of the backtracking procedure constitute a tree-shaped proof of this fact.

A moment of thought reveals that the constructed proof is (isomorphic to) a bivalent RAA proof. Thus backtracking can be seen as a proof search algorithm for $\mathcal{RAA}_B$ as defined in definition 4.29 on page 70. Note, however, that searching exhaustively for proofs with back-tracking can be very expensive in terms of the shortest RAA proof, and this proof may in turn be long and/or deep compared to the shortest dilemma proof. Thus, backtracking is not designed to be an efficient general proof method. Rather, it is an algorithm that we can apply when for example 1-saturation has given us enough information about a formula (in terms of equivalences and implications) to make us believe that we can heuristically choose a variable ordering which will lead us to a proof faster than the more undirected "brute-force" search of κ-saturation.

Extension to Finite Domain Integer Arithmetic

The proof systems and proof search algorithms described in this thesis are all defined for propositional logic. In applications, it is sometimes necessary, or at least very convenient, to be able to express arithmetic relations between numbers. In Prover Plug-in™, the propositional logic language is augmented with functionality for integer arithmetic and the proof search algorithm used is an extension of Stålmarck's method to finite domain integer arithmetic. Formulas can also declare and use enumerated types, which is useful for system modelling.

Stålmarck's method has also been extended in various other ways, for example to many sorted first order logic and to propositional linear temporal logic. As before, we refer to `www.prover.com` for an updated list of references.

Chapter 5

Tool kit

This chapter is devoted to developing some of the tools used to prove the results in chapter 6.

5.1 Simulations and Separations

In order to be able to compare proof systems in more detail, we generalize the concept of p-simulation in definition 2.13 as follows.

Definition 5.1 (Simulation) *Suppose that $\mathcal{P}_1$ and $\mathcal{P}_2$ are propositional proof systems and that M is some proof-theoretic measure defined on both $\mathcal{P}_1$ and $\mathcal{P}_2$.*

$\mathcal{P}_1$ is said to simulate $\mathcal{P}_2$ with respect to M *if there exists a polynomial-time computable function f mapping proofs π in $\mathcal{P}_2$ to equivalent proofs $f(\pi)$ in $\mathcal{P}_1$ such that $M_{\mathcal{P}_2}(f(\pi)) \leq M_{\mathcal{P}_1}(\pi)$.*

The proof system $\mathcal{P}_1$ simulates $\mathcal{P}_2$ linearly *with respect to M if the function f mapping proofs from $\mathcal{P}_2$ to $\mathcal{P}_1$ satisfies $M_{\mathcal{P}_2}(f(\pi)) \leq \mathrm{O}(M_{\mathcal{P}_1}(\pi))$.*

If there exists a polynomial-time computable function f from $\mathcal{P}_2$ to $\mathcal{P}_1$ such that $M_{\mathcal{P}_1}(f(\pi)) \leq M_{\mathcal{P}_2}(\pi)^{\mathrm{O}(1)}$, we say that $\mathcal{P}_1$ simulates $\mathcal{P}_2$ polynomially *(or* p-simulates $\mathcal{P}_2$*) with respect to M.*

Two proof systems can be shown to be "essentially equally strong" with respect to some measure M by proving that they simulate each other (usually polynomially, but that depends on the measure) with respect to M. We demonstrate that two proof systems $\mathcal{P}_1$ and $\mathcal{P}_2$ are *not* "essentially equally strong" by finding a family of (polynomial-size) formulas which separates them with respect to some measure M. If we find such a family of formulas F_n which shows that $\mathcal{P}_1$ is stronger than $\mathcal{P}_2$, we say that the formulas F_n *separate $\mathcal{P}_1$ from $\mathcal{P}_2$* (with respect to M).

Definition 5.2 (Separation) *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be propositional proof systems and M be some measure defined on both $\mathcal{P}_1$ and $\mathcal{P}_2$. Suppose that $\{F_n\}_{n=1}^{\infty}$ is a family of polynomial-size formulas (with $S(F_n) = \omega(1)$) and let π_i^n denote a minimal proof for F_n in $\mathcal{P}_i$ with respect to M.*

If $M_{\mathcal{P}_1}(\pi_1^n) = \mathrm{O}(1)$, the formula family F_n is a logarithmic separation *of $\mathcal{P}_1$ from $\mathcal{P}_2$ with respect to M if $M_{\mathcal{P}_2}(\pi_2^n) = \Omega(\log n)$, a* superlogarithmic separation *if $M_{\mathcal{P}_2}(\pi_2^n) = \omega(\log n)$ and a* linear separation *if $M_{\mathcal{P}_2}(\pi_2^n) = \Omega(n)$.*

If $M_{\mathcal{P}_1}(\pi_1^n) = \Omega(n)$ and there is a function $g : \mathbb{N} \mapsto \mathbb{N}$ such that $M_{\mathcal{P}_2}(\pi_2^n) \geq g(M_{\mathcal{P}_1}(\pi_1^n))$, we say that F_n is a superpolynomial separation *of $\mathcal{P}_1$ from $\mathcal{P}_2$ with respect to M if $g(m) = \omega(m^k)$ for all $k \in \mathbb{N}$ and an* exponential separation *if $g(m) = \Omega(\exp m^c)$ for some $c \in \mathbb{R}^+$.*

Remark 5.3 Note that in order to prove a linear separation of $\mathcal{P}_1$ from $\mathcal{P}_2$, it suffices to find a formula family $\{F_n\}_{n=1}^{\infty}$ with $M_{\mathcal{P}_2}(\pi_2^n) = \Omega(n^c)$ for some $c \in \mathbb{R}^+$ (set $G_n := F_{n^{\lceil 1/c \rceil}}$ to get a polynomial-size linear separation $\{G_n\}_{n=1}^{\infty}$). In particular, it is sufficient to find formulas F_n with $M_{\mathcal{P}_2}(\pi_2^n) = \Omega(n/\log n)$.

Definition 5.4 *We say that $\mathcal{P}_1$ is (linearly, superpolynomially or exponentially)* stronger than $\mathcal{P}_2$ with respect to M *if $\mathcal{P}_1$ simulates $\mathcal{P}_2$ (in absolute terms, polynomially or polynomially, respectively) but the two systems can be separated with respect to M (linearly, superpolynomially or exponentially, respectively).*

If there are separations both ways (of a relevant kind) for the measure M in question, $\mathcal{P}_1$ and $\mathcal{P}_2$ are said to be incomparable with respect to M.

Not all kinds of separations are relevant for all measures. For proof length and size, for example, usually only superpolynomial or exponential separations are of interest. For hardness degree, the interesting types of separations are superlogarithmic and linear ones.

5.2 Modifications of Dilemma

In this section we present two modifications of the dilemma proof system which are of little interest in their own right but will be useful when proving theorems about dilemma and reductio and their relation to resolution.

5.2.1 Normal Form Dilemma

The standard way of showing lower bounds on different proof-theoretic measures in a propositional proof system $\mathcal{P}$ is to study minimal proofs for formula families F_n in $\mathcal{P}$. Often such formula families F_n are conveniently defined by allowing connectives with arity depending on n.

Formally speaking, formulas with connectives of arbitrary arity are not in propositional logic as defined in definition 2.4, so in order for the results concerning them to be valid they should be parenthesized in some way to make the connectives binary. When showing the lower bounds, however, it can be advantageous to avoid arbitrary parenthesizing and instead exploit symmetries in the formulas to simplify the proofs. With this in mind, we present an extension of dilemma to formulas over the set of connectives $\{\neg, \wedge, \vee\}$ where conjunction and disjunction have arbitrary arity but negation can be applied on atomic variables only. More formally, assuming (as before) the existence of a set of variables *Vars*, we make the following definition.

Definition 5.5 (Extended propositional logic) *Let the set of literals Lit be defined by $Lit := \{x, \overline{x} \mid x \in Vars\}$ and let Conj and Disj be the smallest sets such that*

1. *$a \in Conj$ and $a \in Disj$ for all $a \in Lit$,*

2. *if* $P_i \in Disj$ *for* $i = 1, \ldots, n$ *(where* $n \geq 2$*), then* $\bigwedge_{i=1}^{n} P_i \in Conj$*,*

3. *if* $P_i \in Conj$ *for* $i = 1, \ldots, n$ *(where* $n \geq 2$*), then* $\bigvee_{i=1}^{n} P_i \in Disj$*.*

Then the set of "extended" propositional logic formulas $PROP_{ext}$ *is defined as*

$$PROP_{ext} := Conj \cup Disj.$$

Note that the elements in *Conj* and *Disj* are defined as sets, so that for instance the formulas $a \wedge b \wedge c$, $a \wedge c \wedge b$ and $a \wedge b \wedge c \wedge b \wedge a$ are all considered identical. Note also that both CNF and DNF formulas are in $PROP_{ext}$.

We will use the term *extended* or *normal form dilemma* (NF-dilemma for short) to refer to the dilemma proof system adapted to $PROP_{ext}$. When we need to make the distinction, we will use the term *binary dilemma* for dilemma proof system variants with ordinary binary logical connectives.

The propagation rules in NF-dilemma are basically the same as those for binary dilemma. We list the rules in figures 5.1 and 5.2 on the following page (compare with figures A.2 and A.3 in appendix A).

While we can leave the propagation rules basically unaltered, we have to be more careful with the formula relations. A first try at defining formula relations on formulas in $PROP_{ext}$ would be to include all possible subformulas in the domain. But this is not a good idea. For instance, the formula

$$x \wedge y \wedge z \tag{5.1}$$

would give rise to the set of subformulas

$$\{x \wedge y \wedge z, x \wedge y, x \wedge z, y \wedge z, x, y, z\}, \tag{5.2}$$

and in the worst case, the formula relation domain could be exponential in the size of the formula. This would mean that a short proof (i.e. with few lines) could still be exponentially large (since the formula relation on each line is potentially exponential in size), which is undesirable.

Instead, we start with a relation on a formula F with smallest possible domain, i.e. $\{F, F^C, \top, \bot\}$. In each step of an NF-dilemma derivation, we add to the domain of the formula relation the formulas shown or assumed true, false or equivalent (as well as their complements). Propagation rules can be applied to subformulas currently in the formula relation domain and to parents of such formulas. By reasoning analogous to that in section 4.2.2, these are the only subformulas which can yield new equivalences at that point in the derivation.

As before we may branch over arbitrary subformulas of F. The intersection of two formula relations is defined as

$$\mathsf{R}_1 \sqcap \mathsf{R}_2 := \{\{\mathsf{C}_1 \cap \mathsf{C}_2\} \mid (\mathsf{C}_1, \mathsf{C}_2) \in \mathsf{R}_1 \times \mathsf{R}_2, |\{\mathsf{C}_1 \cap \mathsf{C}_2\}| \geq 2\}. \tag{5.3}$$

From (5.3) it follows that a subformula P introduced in the formula relation domain in one of the branches in a dilemma rule application (i.e. a subformula derived true or false or related to some other subformula) is eliminated by the formula relation intersection if there are no common equivalences $P \equiv Q$ derived in both dilemma branches. In this way, we avoid adding superfluous subformulas to the the formula relation when closing dilemma rule applications and make sure that we keep the domain of the relation as small as possible. In particular, there are no unit equivalence classes $[P]$ in an NF-dilemma formula relation.

$$\frac{\bigwedge_{i=1}^{n} P_i \ \wedge\ \bigwedge_{j=1}^{m} Q_j \equiv \top}{\bigwedge_{i=1}^{n} P_i \equiv \top} \tag{C1$_{NF}$}$$

$$\frac{\bigwedge_{i=1}^{n} P_i \ \wedge\ \bigwedge_{j=1}^{m} Q_j \equiv (\bigwedge_{i=1}^{n} P_i)^C}{\bigwedge_{i=1}^{n} P_i \equiv \top} \tag{C2$_{NF}$}$$

$$\frac{\bigwedge_{i=1}^{n} P_i \ \wedge\ \bigwedge_{j=1}^{m} Q_j \equiv (\bigwedge_{i=1}^{n} P_i)^C}{\bigwedge_{j=1}^{m} Q_j \equiv \bot} \tag{C3$_{NF}$}$$

$$\frac{\bigwedge_{i=1}^{n} P_i \equiv \top}{\bigwedge_{i=1}^{n} P_i \ \wedge\ \bigwedge_{j=1}^{m} Q_j \equiv \bigwedge_{j=1}^{m} Q_j} \tag{C4$_{NF}$}$$

$$\frac{\bigwedge_{i=1}^{n} P_i \equiv \bot}{\bigwedge_{i=1}^{n} P_i \ \wedge\ \bigwedge_{j=1}^{m} Q_j \equiv \bot} \tag{C5$_{NF}$}$$

$$\frac{\bigwedge_{i=1}^{n} P_i \equiv \bigwedge_{j=1}^{m} Q_j}{\bigwedge_{i=1}^{n} P_i \ \wedge\ \bigwedge_{j=1}^{m} Q_j \equiv \bigwedge_{i=1}^{n} P_i} \tag{C6$_{NF}$}$$

$$\frac{\bigwedge_{i=1}^{n} P_i \equiv \left(\bigwedge_{j=1}^{m} Q_j\right)^C}{\bigwedge_{i=1}^{n} P_i \ \wedge\ \bigwedge_{j=1}^{m} Q_j \equiv \bot} \tag{C7$_{NF}$}$$

Figure 5.1: NF-dilemma propagation rules for conjunction.

$$\frac{\bigvee_{i=1}^{n} P_i \ \vee\ \bigvee_{j=1}^{m} Q_j \equiv \bot}{\bigvee_{i=1}^{n} P_i \equiv \bot} \tag{D1$_{NF}$}$$

$$\frac{\bigvee_{i=1}^{n} P_i \ \vee\ \bigvee_{j=1}^{m} Q_j \equiv (\bigvee_{i=1}^{n} P_i)^C}{\bigvee_{i=1}^{n} P_i \equiv \bot} \tag{D2$_{NF}$}$$

$$\frac{\bigvee_{i=1}^{n} P_i \ \vee\ \bigvee_{j=1}^{m} Q_j \equiv (\bigvee_{i=1}^{n} P_i)^C}{\bigvee_{j=1}^{m} Q_j \equiv \top} \tag{D3$_{NF}$}$$

$$\frac{\bigvee_{i=1}^{n} P_i \equiv \top}{\bigvee_{i=1}^{n} P_i \ \vee\ \bigvee_{j=1}^{m} Q_j \equiv \top} \tag{D4$_{NF}$}$$

$$\frac{\bigvee_{i=1}^{n} P_i \equiv \bot}{\bigvee_{i=1}^{n} P_i \ \vee\ \bigvee_{j=1}^{m} Q_j \equiv \bigvee_{j=1}^{m} Q_j} \tag{D5$_{NF}$}$$

$$\frac{\bigvee_{i=1}^{n} P_i \equiv \bigvee_{j=1}^{m} Q_j}{\bigvee_{i=1}^{n} P_i \ \vee\ \bigvee_{j=1}^{m} Q_j \equiv \bigvee_{i=1}^{n} P_i} \tag{D6$_{NF}$}$$

$$\frac{\bigvee_{i=1}^{n} P_i \equiv \left(\bigvee_{j=1}^{m} Q_j\right)^C}{\bigvee_{i=1}^{n} P_i \ \vee\ \bigvee_{j=1}^{m} Q_j \equiv \top} \tag{D7$_{NF}$}$$

Figure 5.2: NF-dilemma propagation rules for disjunction.

If we define formula relations on formulas in $PROP_{ext}$ in this way, the size of a formula relation will always be polynomial in the length of a derivation, and thus derivation length and size will be polynomially related as well.

The point of introducing NF-dilemma is to simplify the proofs of bounds on dilemma derivations by exploiting the symmetry in $PROP_{ext}$-formulas (and more specifically in CNF and DNF formulas).

Consider for example a contradictory k-CNF formula F parenthesized in some natural way so that the connectives are binary. Any dilemma refutation (atomic, bivalent or general) of F can be regarded as an NF-dilemma refutation. Turning this around, if we can prove that there is no refutation of depth d or length L of F in NF-dilemma, then there is no such refutation in ordinary binary dilemma either. And proving that there is no NF-dilemma proof can be easier, since by using symmetry we can assume without loss of generality that an atomic proof, say, branches on a specific variable without having to worry about how the clauses in the formula are parenthesized.

As a final remark, we observe that as defined above, NF-dilemma lacks one of the most important characteristics of binary dilemma from a practical point of view. Namely, we lose the possibility to search exhaustively for proofs of limited depth in an efficient way. Since there might be an exponential number of branching possibilities when saturating a formula relation, the upper bound for the complexity of proof search in theorem 4.34 on page 77 no longer holds.

5.2.2 Equivalence-Based Dilemma

One of the problems when comparing dilemma and resolution is that if we want to translate a dilemma proof into resolution, we have to take into consideration that all information gathered so far in the dilemma derivation is encoded in the formula relation on the current line in the derivation. This corresponds very poorly to how resolution proofs work. Our next modification of the dilemma proof system is designed specifically to eliminate this problem and thus facilitate the translation of dilemma proofs to resolution.

The solution is to eliminate the usage of formula relations. Instead, we let the propagation rules derive formula equivalences from other formula equivalences occurring earlier in the proof. We do this by introducing the concept of scope and specifying that equivalences derived in the current scope, and only such equivalences, can be used to derive new formula equivalences. Also, we introduce rules for formal manipulation of formula equivalences, such as for instance

$$\frac{P \equiv Q \quad Q \equiv R}{P \equiv R} \tag{5.4}$$

for transitivity and

$$\frac{P \equiv Q}{P^C \equiv Q^C} \tag{5.5}$$

for complement (see figure A.1 on page 126 for the full list of rules for formula equivalences). We replace the canonical explicitly contradictory formula relation by overloading the symbol $\perp$ to denote that a contradiction has been derived and adding the rule

$$\frac{P \equiv P^C}{\perp}. \tag{5.6}$$

The ideas sketched above are developed in more detail and formalized in appendix A.

Any variant of dilemma with formula relations (atomic, bivalent or general variants of binary or even NF-dilemma, and correspondingly for reductio) can be made into a proof system using formal manipulation of formula equivalences. We will call such modifications of the dilemma or reductio proof systems "equivalence-based". From the discussions in section A.5 we get the following proposition.

Proposition 5.6
If F is a tautology, $\mathcal{P}$ is a proof system using formula relations and $\mathcal{P}^{\equiv}$ is the corresponding equivalence-based system, then it holds that

$$L_{\mathcal{P}}(\vdash F) \leq L_{\mathcal{P}^{\equiv}}(\vdash F) \leq L_{\mathcal{P}}(\vdash F)^{\mathrm{O}(1)}$$

and

$$S_{\mathcal{P}}(\vdash F)/\mathrm{O}\left(S\left(F\right)^{2}\right) \leq S_{\mathcal{P}^{\equiv}}(\vdash F) \leq S_{\mathcal{P}}(\vdash F)^{\mathrm{O}(1)}.$$

The hardness degree is the same in the two proof systems, i.e.

$$H_{\mathcal{P}^{\equiv}}\left(F\right) = H_{\mathcal{P}}\left(F\right).$$

In other words, it does not really matter whether we use proof system variants based on formula relations or formula equivalences for the theoretical analysis as long as polynomial differences are considered insignificant.

5.3 Lower Bounds and Proof Hardness

When defining a measure of proof hardness $H_{\mathcal{P}}$ in a proof system $\mathcal{P}$, we want it to capture the intuitive concept that a propositional logic formula F is hard if and only if there are no small (or short) proofs of F. More formally, if $\{F_n\}_{n=1}^{\infty}$ is a family of polynomial-size tautologies we want the measure $H_{\mathcal{P}}$ to possess the properties

$$H_{\mathcal{P}}\left(F_n\right) = \Omega\left(n\right) \;\Rightarrow\; S_{\mathcal{P}}(\vdash F_n) = \exp\bigl(\Omega\left(n\right)\bigr) \tag{5.7}$$

and

$$S_{\mathcal{P}}(\vdash F_n) = \exp\bigl(\Omega\left(n\right)\bigr) \;\Rightarrow\; H_{\mathcal{P}}\left(F_n\right) = \Omega\left(n^{c}\right) \tag{5.8}$$

for some $c \in \mathbb{R}^{+}$.

All dilemma and RAA proof systems satisfy the property (5.8) that F must be hard if there are no small proofs of F.

Theorem 5.7 (Size implies hardness for dilemma and RAA systems)
Let $\{F_n\}_{n=1}^{\infty}$ be a family of polynomial-size tautologies and let $\mathcal{P}$ denote one of the dilemma proof systems $\mathcal{D}$, $\mathcal{D}_B$ or $\mathcal{D}_A$ or one of the reductio systems $\mathcal{RAA}$, $\mathcal{RAA}_B$ or $\mathcal{RAA}_A$. Then it holds that if

$$S_{\mathcal{P}}(\vdash F_n) = \exp\bigl(\Omega\left(n\right)\bigr)$$

then

$$H_{\mathcal{P}}\left(F\right) = \Omega\left(n/\log n\right).$$

Proof: By the inequalities (4.20) on page 51 we have

$$L_{\mathcal{P}}(\vdash F_n) = \Omega\left(S_{\mathcal{P}}(\vdash F_n) \,/\, S(F_n)^2\right). \tag{5.9}$$

An inspection of theorem 4.23 on page 58 reveals that the proof of the upper bound is independent of the dilemma or reductio system used, so

$$L_{\mathcal{P}}(\vdash F_n) = \mathrm{O}\left(S(F_n)^{H_{\mathcal{P}}(F_n)+1}\right). \tag{5.10}$$

Since $S(F_n)$ is polynomial in n and $S_{\mathcal{P}}(\vdash F_n)$ is exponential in n, it follows from (5.9) and (5.10) that $H_{\mathcal{P}}(F_n) = \Omega(n/\log n)$. □

In the rest of this section, we discuss the other direction for the dilemma and RAA proof systems, i.e. whether the property (5.7) holds and proof hardness implies proof size, and give strategies for how to prove lower bounds on hardness.

A natural approach to proving bounds on hardness is as follows. Given a family of propositional logic formulas $\{F_n\}_{n=1}^{\infty}$, define a family of (implicitly contradictory) formula relations $\{\mathsf{R}_n\}_{n=1}^{\infty}$, where each R_i corresponds to $F_{g(i)}$ for some function $g(n) = \Omega(n)$. Then prove that the hardness of the formulas F_n grows as $g^{-1}(n)$ by induction by showing $H_{\mathcal{P}}(\mathsf{R}_i) \geq H_{\mathcal{P}}(\mathsf{R}_{i-1}) + 1$.

For RAA proof systems, it is sufficient to show that $\mathsf{R}_i[\psi]$ is at least as hard as R_{i-1} for all "meaningful" assumptions ψ.

Definition 5.8 (Nontrivial association) *Suppose that ψ is an association on a formula relation R. We say that ψ is a* nontrivial association *on R if $\mathsf{R}[\psi] \not\sqsubseteq \mathit{Sat}(\mathsf{R},0)$ and $\mathsf{R}[\psi^C] \not\sqsubseteq \mathit{Sat}(\mathsf{R},0)$. Otherwise ψ is* trivial.

That is, for a trivial association ψ on R, there is no use branching on ψ and try to refute (or derive common associations for) $\mathsf{R}[\psi]$ and $\mathsf{R}[\psi^C]$.

Proposition 5.9 (Obtaining lower bounds on RAA hardness) *Let R_n be a family of implicitly contradictory formula relations and let $\mathcal{P}$ denote one of the reductio proof systems $\mathcal{RAA}$, $\mathcal{RAA}_B$ or $\mathcal{RAA}_A$. If it holds that $H_{\mathcal{P}}(\mathsf{R}_1) \geq 1$ and that*

$$\min\left\{H_{\mathcal{P}}\left(\mathit{Sat}(\mathsf{R}_n,0)[\psi]\right)\right\} \geq H_{\mathcal{P}}(\mathsf{R}_{n-1})$$

for $n > 1$, where the minimum is taken over all nontrivial associations ψ on the relation R_n allowable as assumptions by the branching rules of $\mathcal{P}$, then $H_{\mathcal{P}}(\mathsf{R}_n) = \Omega(n)$.

The proof, which is an easy induction over n, is omitted.

For all (analytic) RAA proof systems, linear growth of hardness implies exponential growth of proof length and size. This is not a very deep result, but we state it as a theorem for reference.

Theorem 5.10 (Hardness implies length for all RAA proof systems) *Suppose that R is an implicitly contradictory formula relations and let $\mathcal{P}$ denote one of the reductio proof systems $\mathcal{RAA}$, $\mathcal{RAA}_B$ or $\mathcal{RAA}_A$. Then*

$$L_{\mathcal{P}}(\mathsf{R} \vdash \bot) \geq 2^{H_{\mathcal{P}}(\mathsf{R})}.$$

Sketch of proof: Show the contrapositive that if there is a $\mathcal{P}$-derivation $\pi : \mathsf{R} \Rightarrow \bot_{\mathsf{R}}$ in length $L(\pi) < b$, then $H_{\mathcal{P}}(\mathsf{R}) < \log b$. Just mimic the proof of theorem 4.18 and simplify the induction step (lemma 4.19 is not needed). $\Box$

Taken together, proposition 5.9 and theorem 5.10 say that the measure of proof hardness in reductio systems have the desired properties (5.7) and (5.8) and provide a strategy for obtaining lower bounds on proof length and size by proving bounds on hardness.

By theorems 4.18 and 4.23, in general dilemma $\mathcal{D}$, too, the hardness measure $H_{\mathcal{D}}$ satisfies (5.7) and (5.8). Moreover, if we can prove a lower bound on $H_{\mathcal{RAA}}(\mathsf{R}_n)$, then the same asymptotic lower bound holds in general dilemma, since $H_{\mathcal{RAA}}(F) \leq 2 \cdot H_{\mathcal{D}}(F)$ (theorem 6.12. in section 6.2).

For the dilemma subsystems $\mathcal{D}_B$ and $\mathcal{D}_A$ the matter is more complicated. Anticipating the results in chapter 6, lower bounds on hardness in the two systems cannot be obtained from bounds in the corresponding RAA systems $\mathcal{RAA}_B$ and $\mathcal{RAA}_A$, so we have to develop other tools for proving bounds on hardness in $\mathcal{D}_B$ and $\mathcal{D}_A$. Also, we have to study the question whether proof hardness implies proof length and size (i.e. whether the lower bound for $\mathcal{D}$ in theorem 4.18 can be translated into bound for the subsystems $\mathcal{D}_B$ and $\mathcal{D}_A$). Below, we discuss how lower bounds on hardness can be obtained in bivalent and atomic dilemma. We defer the question whether proof hardness implies proof length and size in $\mathcal{D}_B$ and $\mathcal{D}_A$ to section 6.1.

Let R_n be a family of (implicitly contradictory) formula relations and let $\mathcal{P}$ denote one of the dilemma proof systems $\mathcal{D}_B$ or $\mathcal{D}_A$. Assume, for simplicity, that all R_n are 0-saturated and suppose that π is a $\mathcal{P}$-refutation of R_n. Then π can be depicted schematically as

$$\begin{array}{c} \mathsf{R}_n \\ \hline \begin{array}{cc} \mathsf{R}[\psi] & \mathsf{R}[\psi^C] \\ \pi_1 & \pi_2 \\ \mathsf{R}_n^1 & \mathsf{R}_n^2 \end{array} \\ \hline \mathsf{R}_n^1 \sqcap \mathsf{R}_n^2 \\ \pi_3 \\ \bot_{\mathsf{R}_n} \end{array} \tag{5.11}$$

(where we might have $\mathsf{R}_n^1 \sqcap \mathsf{R}_n^2 = \bot_{\mathsf{R}_n}$, in which case π_3 is superfluous).

Just as for reductio, we would like to show that the hardness degree grows at least linearly with n by proving that $H_{\mathcal{P}}(\mathsf{R}_n[\psi]) \geq H_{\mathcal{P}}(\mathsf{R}_{n-1})$ for all nontrivial associations ψ on the relation R_n allowable as assumptions by the branching rules of $\mathcal{P}$. But this is not enough. The above inequality does not exclude the possibility that there are subderivations $\pi_1 : \mathsf{R}[\psi] \Rightarrow \mathsf{R}_n^1$ and $\pi_2 : \mathsf{R}[\psi^C] \Rightarrow \mathsf{R}_n^2$ of depth strictly less than $H_{\mathcal{P}}(\mathsf{R}_{n-1})$ which can be inserted in (5.11) to derive common equivalences that reduce R_n to a formula relation $\mathsf{R}_n^1 \sqcap \mathsf{R}_n^2$ with hardness degree $H_{\mathcal{P}}(\mathsf{R}_n^1 \sqcap \mathsf{R}_n^2) = H_{\mathcal{P}}(\mathsf{R}_{n-1})$.

What we need, therefore, is a stronger result. For example, we can try to show by induction that if $\max\{D(\pi_1), D(\pi_2)\} < n-1$ in the derivation (5.11), then it must hold that $\mathsf{R}_n^1 \sqcap \mathsf{R}_n^2 = \mathsf{R}_n$ (i.e. that no new equivalences whatsoever can be derived from the relation R_n in depth less than n). A strategy for such a proof is outlined in the next proposition. We remind the reader that a proper nontrivial dilemma derivation $\pi : \mathsf{R}_1 \Rightarrow \mathsf{R}_2$ is a derivation without redundant derivation steps and with $\mathsf{R}_1 \neq \mathsf{R}_2$ (definition 4.10).

Proposition 5.11 (Obtaining lower bounds on dilemma hardness) *Let R_n be a family of implicitly contradictory formula relations and let $\mathcal{P}$ denote one of the dilemma proof systems $\mathcal{D}_B$ or $\mathcal{D}_A$.*

Suppose that the following holds:

1. $H_{\mathcal{P}}(\mathsf{R}_n) \geq 1$ *for all* $n \in \mathbb{N}$,
2. *for all nontrivial associations* ψ *on the relation* R_n, $n \geq 2$, *allowable as assumptions by the branching rules of* $\mathcal{P}$, *it holds that*
$$Sat\big(\mathsf{R}_n[\psi], 0\big) \sqcap Sat\big(\mathsf{R}_n[\psi^C], 0\big) = Sat\big(\mathsf{R}_n, 0\big),$$
3. *for all nontrivial associations* ψ *on* R_n, $n \geq 2$, *allowable as assumptions by the branching rules of* $\mathcal{P}$, *the relation* $\mathsf{R}_n[\psi]$ *is as least as difficult as* R_{n-1} *in the following sense:*

   If there exists a proper nontrivial $\mathcal{P}$*-derivation* $\pi : Sat\big(\mathsf{R}_n[\psi], 0\big) \Rightarrow \mathsf{R}'_n$ *in depth* $D(\pi) \leq n-1$, *then* π *can be modified to yield a proper nontrivial* $\mathcal{P}$*-derivation* $\pi' : Sat\big(\mathsf{R}_{n-1}, 0\big) \Rightarrow \mathsf{R}'_{n-1}$ *in depth* $D(\pi') \leq D(\pi)$.

Then for all $n \in \mathbb{N}$ *it must hold that if* $\pi : Sat\big(\mathsf{R}_n, 0\big) \Rightarrow \mathsf{R}'_n$ *is a proper nontrivial* $\mathcal{P}$*-derivation (and such derivations exist), then* $D(\pi) \geq n$. *In particular, it holds that*
$$H_{\mathcal{P}}(\mathsf{R}_n) = \Omega(n).$$

Proof: By induction over n.

The base case $n = 1$ follows by definition.

For $n \geq 2$, R_n is 1-hard by condition 1, so certainly there exist proper nontrivial dilemma derivations $\pi : Sat\big(\mathsf{R}_n, 0\big) \Rightarrow \mathsf{R}'_n$. Suppose that π is such a derivation. The first step in π must be a dilemma rule application with subderivations $\pi_1 : Sat\big(\mathsf{R}_n, 0\big)[\psi] \Rightarrow \mathsf{R}_1$ and $\pi_2 : Sat\big(\mathsf{R}_n, 0\big)[\psi^C] \Rightarrow \mathsf{R}_2$ for some nontrivial association ψ.

By condition 2, 0-saturating $\mathsf{R}_n[\psi]$ and $\mathsf{R}_n[\psi^C]$ and taking the intersection yields no new equivalences. Since π is a proper derivation, it follows that we must have either $\mathsf{R}_1 \sqsupset Sat\big(\mathsf{R}_n[\psi], 0\big)$ in π_1 or $\mathsf{R}_2 \sqsupset Sat\big(\mathsf{R}_n[\psi^C], 0\big)$ in π_2. Combining condition 3 with the induction hypotesis, we conclude that $\max\{D(\pi_1), D(\pi_2)\} \geq n-1$. The proposition follows. □

So if we can show that a formula relation family R_n satisfies conditions 1, 2 and 3 in proposition 5.11, this proposition yields a linear lower bound on $H(\mathsf{R}_n)$. The problem, though, is how to prove condition 3.

If R_1 and R_2 are formula relations on the same domain with $\mathsf{R}_1 \sqsubseteq \mathsf{R}_2$, then any derivation π from R_1 can be made into a derivation from R_2 in a natural way (just augment π by some set of associations Ψ such that $\mathsf{R}_2 = \mathsf{R}_1[\Psi]$). We would need some kind of tool for showing analogous results when R_2 is stronger than R_1 in a "structural sense" although the formula relations are defined on different domains $Sub(\mathsf{R}_1) \neq Sub(\mathsf{R}_2)$.

There are (at least) two rather obvious suggestions for such tools:

1. In section 2.6, we defined restrictions ρ on CNF formulas F and resolution derivations π (definition 2.31). The most important property of restrictions is the fact that they preserve resolution derivations. That is, if π

is a resolution refutation of F, then $\pi|_\rho$ is a resolution refutation of $F|_\rho$ (proposition 2.33).

It does not seem to be an all too far-fetched idea to try to mimic these definitions for dilemma. First, one would define 1-restrictions for the general class of propositional logic formulas and extend these definitions to arbitrary restrictions ρ on formulas F in the natural way. Then, one would make analogous definitions for formula relations. Finally, one would hopefully be able to prove that if $\pi : F^\top \Rightarrow \bot_F$ is a dilemma refutation of F, then $\pi|_\rho : F^\top|_\rho \Rightarrow \bot_{F|_\rho}$ is a dilemma refutation of $F|_\rho$.

2. A possibly somewhat more general idea would be to formalize the notion of structural similarities between formula relations and define some kind of formula relation homomorphism. A preliminary suggestion would be as follows.

   Let R be a formula relation and let C_i denote equivalence classes in this relation. Recalling the notation in section 4.2.2, we say that the relation $\langle \mathsf{C}_1 : \mathsf{C}_2 \circ \mathsf{C}_3 \rangle$ holds if there exists a subformula $P_1 \doteq P_3 \circ P_3$ in $Sub(\mathsf{R})$ such that $P_i \in \mathsf{C}_i$ for $i = 1, 2, 3$ (where as usual $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$).

   Let R_1 and R_2 be formula relations (not necessarily compatible). A function $f : \mathsf{R}_1 \mapsto \mathsf{R}_2$ is a *formula relation homomorphism* from R_1 to R_2 if the following holds for equivalence classes $\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3 \in \mathsf{R}_1$:

   (a) if $\top \in \mathsf{C}_1$ then $\top \in f(\mathsf{C}_1)$,

   (b) $f(\mathsf{C}_1{}^C) = f(\mathsf{C}_1)^C$,

   (c) if $x \in \mathsf{C}_1$ for some variable $x \in Vars(\mathsf{R}_1)$, then there exists a variable $y \in Vars(\mathsf{R}_2)$ such that $y \in f(\mathsf{C}_1)$,

   (d) if $\langle \mathsf{C}_1 : \mathsf{C}_2 \circ \mathsf{C}_3 \rangle$ in R_1 then $\langle f(\mathsf{C}_1) : f(\mathsf{C}_2) \circ f(\mathsf{C}_3) \rangle$ in R_2.

   That is, determinate equivalence classes are mapped on corresponding determinate equivalence classes, complements are mapped on complements, atomic equivalence classes are mapped on atomic equivalence classes and any relation between equivalence classes in R_1 are preserved in R_2.

   If we can find a formula relation homomorphism $f : \mathsf{R}_1 \mapsto \mathsf{R}_2$, then it should hold that any derivation $\pi_1 : \mathsf{R}_1 \Rightarrow \mathsf{R}_1'$ corresponds to a derivation $\pi_2 : \mathsf{R}_2 \Rightarrow f(\mathsf{R}_1')$ of the same shape (start with $\pi_2 = f(\pi_1)$ and then possibly modify π_2 somewhat to make it proper and standardized). In particular, this would hold for refutations, from which would follow the desired conclusion $H(\mathsf{R}_1) \geq H(\mathsf{R}_2)$.

There are quite a few fine points which need to be taken care of if the ideas about formula relation restrictions or formula relation homomorphisms are to be formalized, however. Therefore, it has not been practically feasible to study these questions in any detail within the framework of this Master's thesis. Instead, we prove our results in chapter 6 concerning bounds on proof hardness in dilemma by developing ad-hoc tools for each separate result and applying proposition 5.11 where possible.

Chapter 6

Results

The moment of truth has finally come. We have arrived at the point where we present the results of our Master's project.

This chapter contains a comparative study of the dilemma, reductio and resolution proof systems, in which the proof power of different modifications of the systems are examined and related to each other. The proof-theoretic measures of interest are proof depth (where applicable), length and size. The ultimate goal of the comparisons is to establish either p-simulations or superpolynomial (or preferably exponential) separations with respect to proof length and/or size between the different proof systems.

While chapters 3 and 4 to a large extent are accounts of earlier research, this chapter consists mainly of our original contributions, based on the tools developed in chapter 5. Results presented in this chapter that have been shown earlier are provided with references. In the cases where no such references are given, the results are, to our knowledge, new.

6.1 Subsystems of Dilemma

We begin our comparative study by examining the relative strength of the atomic ($\mathcal{D}_A$), bivalent ($\mathcal{D}_B$) and general ($\mathcal{D}$) dilemma proof systems.

Plainly, $\mathcal{D}_B$ simulates $\mathcal{D}_A$ and $\mathcal{D}$ simulates $\mathcal{D}_B$ with respect to any measure M, since $\mathcal{D}_A \subseteq \mathcal{D}_B \subseteq \mathcal{D}$ as proof systems. We are interested in what relations hold in the other direction.

Our first result is that $\mathcal{D}_B$ is linearly stronger than $\mathcal{D}_A$ with respect to hardness degree.

Theorem 6.1 (Linear separation of $\mathcal{D}_B$ from $\mathcal{D}_A$ w.r.t. hardness)
There is a family of polynomial-size propositional logic formulas F_n which separates bivalent dilemma from atomic dilemma linearly with respect to hardness degree (i.e. such that $H_{\mathcal{D}_B}(F_n) = \mathrm{O}(1)$ but $H_{\mathcal{D}_A}(F_n) = \Omega(n)$).

Sketch of proof: Consider the formula family CM_n in figure 6.1 on the following page. For a fix n, CM_n is the (obviously contradictory) formula saying that it is possible to fill in the cells in an n by n matrix with zeroes and ones in such a way that there is both a row consisting only of ones and a column consisting only of zeroes. We claim that the formulas CM_n (parenthesized in

$$R_i^n := \bigwedge_{1 \le j \le n} x_{i,j} \tag{6.1}$$

$$C_j^n := \bigwedge_{1 \le i \le n} \overline{x}_{i,j} \tag{6.2}$$

$$CM_n := \bigvee_{1 \le i \le n} R_i^n \wedge \bigvee_{1 \le j \le n} C_j^n \tag{6.3}$$

Figure 6.1: Contradictory matrix formula CM_n.

some suitable way to get formulas in propositional logic with binary connectives in accordance with definition 2.4) is a linear separation with respect to hardness of $\mathcal{D}_B$ from $\mathcal{D}_A$.

Showing that $H_{\mathcal{D}_B}(F_n) = O(1)$ is easy. To get a bivalent dilemma refutation (indeed, a bivalent RAA refutation) of CM_n in depth 1, assume CM_n true and start by branching over R_1^n. If $R_1^n \equiv \top$, then $C_j^n \equiv \bot$ for all j and we get a contradiction. Hence, $R_1^n \equiv \bot$. We repeat this for $i = 2, \ldots, n$. But if all R_i^n are false, so is CM_n. Contradiction.

Strictly speaking, this is not a binary refutation but a normal form refutation of an extended propositional logic formula (as defined in section 5.2.1). It is immediate, however, that if we parenthesize CM_n in any natural way to get a (binary) propositional logic formula CM'_n, the proof described above forms the backbone of a 1-refutation of CM'_n in binary bivalent reductio.

Proving the linear lower bound on $H_{\mathcal{D}_A}(F_n)$ is considerably harder. We only sketch the idea behind the proof below and refer to section B.1 for a detailed proof.

By section 5.2.1, it suffices to give a lower bound on hardness in atomic NF-dilemma of the (extended propositional logic) formulas CM_n, since such a bound must also hold in binary dilemma. So suppose that π is an atomic NF-dilemma refutation of CM_n. It is easy to see that $H_{\mathcal{D}_A}(F_n) \ge 1$ for $n \ge 2$, so π must contain a dilemma rule application branching over some variable $x_{i,j}$. Because of symmetry, we may without loss of generality assume $i = j = n$.

Assuming $x_{n,n} \equiv \top$ falsifies C_n^n. The complementary assumption $x_{n,n} \equiv \bot$ falsifies R_n^n. This is essentially everything of interest that can be derived in the two branches by 0-saturation, and since there are no common new associations there must be nested dilemma rule applications in at least one of the branches. It seems intuitively plausible that deriving any further consequences in the branch assuming $x_{n,n} \equiv \top$ is just as hard as refuting CM_n with the last column deleted, and analogously with the last row deleted in the branch assuming $x_{n,n} \equiv \bot$. By an appeal to proposition 5.11, we get the lower bound $H_{\mathcal{D}_A}(F_n) = \Omega(n)$. $\square$

While it is not very surprising that $\mathcal{D}_B$ is linearly stronger than $\mathcal{D}_A$ with respect to hardness for general propositional logic formulas, it is not obvious that the same should hold also for CNF formulas. On the one hand, a dilemma rule assumption of type $\bigvee_{i=1}^n a_i \equiv \bot$ is strong, since it immediately follows than $a_i \equiv \bot$ for all $i = 1, \ldots, n$. On the other hand, the complementary assumption $\bigvee_{i=1}^n a_i \equiv \top$ is an extremely weak one (unless $n = 1$), so we cannot expect to get anything much out of the branch assuming the disjunction true.

$$A(i,j,k) := \overline{x}_{i,j} \vee \overline{x}_{j,k} \vee x_{i,k} \tag{6.4}$$

$$B(i,j) := \overline{x}_{i,j} \vee \overline{x}_{j,i} \tag{6.5}$$

$$C_n(j) := \bigvee_{\substack{1 \le i \le n \\ i \ne j}} x_{i,j} \tag{6.6}$$

$$A_n := \bigwedge_{1 \le i \le n} \bigwedge_{\substack{1 \le j \le n \\ i \ne j}} \bigwedge_{\substack{1 \le k \le n \\ i \ne k \ne j}} A(i,j,k) \tag{6.7}$$

$$B_n := \bigwedge_{1 \le i \le n-1} \bigwedge_{i < j \le n} B(i,j) \tag{6.8}$$

$$C_n := \bigwedge_{1 \le j \le n} C_n(j) \tag{6.9}$$

$$GT_n := A_n \wedge B_n \wedge C_n \tag{6.10}$$

Figure 6.2: Graph tautology formula GT_n.

Nevertheless, it does seem to be the case that (normal form) bivalent dilemma is linearly stronger than atomic dilemma with respect to hardness even when restricted to CNF formulas. Since it has not been possible to develop the necessary theoretic tools for proving lower bounds in dilemma proof systems within the framework of this Master's thesis (see remarks at the end of section 5.3), we state this as a conjecture and give an argument why we are convinced that this conjecture is true.

Conjecture 6.2 (Separation of $\mathcal{D}_B$ from $\mathcal{D}_A$ w.r.t. hardness for CNF) *There is a family of polynomial-size CNF formulas F_n which separates bivalent NF-dilemma from atomic NF-dilemma linearly with respect to proof hardness.*

Plausibility argument: Our conjectured separation of bivalent dilemma from atomic dilemma for CNF formulas is a formula family encoding the negation of the statement that any strict order over a finite set has a minimal element. If we consider the directed graph associated to a strict order, this statement is equivalent to the following: "Each directed graph closed under transitivity and without loops must have a source node."

The CNF formula GT_n in figure 6.2 expresses the negation of the above statement. The intended meaning of the variables $x_{i,j}$ for $1 \le i,j \le n$, $i \ne j$, is that (i,j) is a directed edge in a graph with vertices $[n]$ if $x_{i,j}$ is true. The subformulas A_n specify that the graph is transitive, the property that there is no 2-cycles in the graph is described by B_n and the statement that the graph has no source node is translated to the clauses C_n.

The formulas GT_n can be proven 1-easy in bivalent reductio by an inductive argument adapted from [54].[1] Obviously,

$$GT_2 = (\overline{x}_{1,2} \vee \overline{x}_{2,1}) \wedge x_{1,2} \wedge x_{2,1} \tag{6.11}$$

is 0-easy. Suppose that GT_n has been shown 1-easy for bivalent reductio and

[1] In fact, the resolution proof in [54] was constructed on the basis of a dilemma proof found by Stålmarck's method [51], and the reductio proof presented here is a retranslation back to dilemma of this resolution proof.

$$
\cfrac{\begin{array}{c} GT_{n+1} \equiv \top \\ A_{n+1} \equiv \top \\ B_{n+1} \equiv \top \\ C_{n+1} \equiv \top \end{array}}{\begin{array}{l|ll} C_n(j) \equiv \top & C_n(j) \equiv \bot & \\ & x_{i,j} \equiv \bot & \{1 \le i \le n,\, i \ne j\} \\ & C_{n+1}(j) \equiv \top & \\ & x_{n+1,j} \equiv \top & \\ & A(i, n+1, j) \equiv \top & \{1 \le i \le n,\, i \ne j\} \\ & x_{i,n+1} \equiv \bot & \{1 \le i \le n,\, i \ne j\} \\ & B(j, n+1) \equiv \top & \\ & x_{j,n+1} \equiv \bot & \\ & C_{n+1}(n+1) \equiv \bot & \\ & C_{n+1}(n+1) \equiv \top & \end{array}}
$$
$$C_n(j) \equiv \top$$

Figure 6.3: $\mathcal{RAA}_B$-derivation of $C_n(j)$ from GT_{n+1} in depth 1.

consider GT_{n+1}. We claim that from the assumption $GT_{n+1} \equiv \top$ we can derive $C_n(j) \equiv \top$ for all $j = 1, \ldots, n$ by (closed) bivalent reductio derivations in depth 1. Since $A_n \subseteq A_{n+1}$ and $B_n \subseteq B_{n+1}$, it follows by induction that GT_{n+1} is 1-easy for bivalent reductio.

It remains to prove the claim. An outline of a 1-depth $\mathcal{RAA}_B$-derivation of $C_n(j)$ from GT_{n+1} is given in figure 6.3, where the notation

$$D_i \equiv \nu \quad \{1 \le i \le n,\, i \ne j\} \tag{6.12}$$

is shorthand for

$$\begin{array}{c} D_1 \equiv \nu \\ \vdots \\ D_{j-1} \equiv \nu \\ D_{j+1} \equiv \nu \\ \vdots \\ D_n \equiv \nu. \end{array} \tag{6.13}$$

In words, suppose that $C_n(j)$ is false, i.e. that there is no edge into vertex j from any of the other n first vertices. Then the graph must have an edge $(n+1, j)$. Since the graph is transitive and has no 2-cycles, it follows that vertex $n+1$ is a source. Contradiction. Consequently $C_n(j)$ must be true.

For atomic dilemma, it is easy to see that $H_{\mathcal{D}_A}(GT_n) \ge 1$ for $n \ge 3$. Suppose (without loss of generality because of symmetry) that an atomic dilemma derivation branches over $x_{n-1,n}$. It is a routine matter (though tedious) to show that

$$\begin{aligned} Sat\big(GT_n{}^\top[x_{n-1,n} \equiv \top], 0\big) \sqcap Sat\big(GT_n{}^\top[x_{n-1,n} \equiv \bot], 0\big) \\ = Sat\big(GT_n{}^\top, 0\big). \end{aligned} \tag{6.14}$$

Thus the first two conditions in proposition 5.11 are met. If one inspects the formula relations $Sat\left(GT_n{}^{\top}\left[x_{n-1,n} \equiv \top\right], 0\right)$ and $Sat\left(GT_n{}^{\top}\left[x_{n-1,n} \equiv \bot\right], 0\right)$, it seems obvious that condition 3 holds too, i.e. that deriving anything from these relations must be at least as hard as deriving nontrivial consequences from GT_{n-1}. If this claim could be proven true, proposition 5.11 would yield $H_{\mathcal{D}_A}(GT_n) = n-2$ (it is easy to rewrite the derivation in figure 6.3 to an atomic derivation in depth $n-2$, so $H_{\mathcal{D}_A}(GT_n) \leq n-2$).

And indeed, tests with the program `prove`, which is an implementation for research purposes of Stålmarck's method, on GT_n for $n = 2, 3, \ldots, 7$ confirm that $H_{\mathcal{D}_A}(GT_n) = n-2$ for these values of n. Constructing an analogue of the laborious and rather clumsy proof in section B.1 of the lower bound in theorem 6.1 has been judged outside the scope of this already rather extensive report, however, wherefore this separation result is left as a conjecture. □

It is trivial to give bounds for the relation between the hardness degrees in $\mathcal{D}$ and $\mathcal{D}_B$. Any general dilemma derivation

$$\begin{array}{c} \mathsf{R} \\ \hline \begin{array}{cc} \mathsf{R}\left[P \equiv Q\right] & \mathsf{R}\left[P \equiv Q^C\right] \\ \pi_1 & \pi_2 \\ \mathsf{R}_1 & \mathsf{R}_2 \end{array} \\ \hline \mathsf{R}_1 \sqcap \mathsf{R}_2 \end{array} \tag{6.15}$$

in depth 1 can be transformed to an equivalent bivalent derivation

$$\begin{array}{c} \mathsf{R} \\ \hline \begin{array}{cc} \begin{array}{c} \mathsf{R}\left[P \equiv \bot\right] \\ \hline \begin{array}{cc} \mathsf{R}\left[Q \equiv \bot\right] & \mathsf{R}\left[Q \equiv \top\right] \\ \pi_1 & \pi_2 \\ \mathsf{R}_1 & \mathsf{R}_2 \end{array} \\ \hline \mathsf{R}_1 \sqcap \mathsf{R}_2 \end{array} & \begin{array}{c} \mathsf{R}\left[P \equiv \top\right] \\ \hline \begin{array}{cc} \mathsf{R}\left[Q \equiv \bot\right] & \mathsf{R}\left[Q \equiv \top\right] \\ \pi_2 & \pi_1 \\ \mathsf{R}_2 & \mathsf{R}_1 \end{array} \\ \hline \mathsf{R}_1 \sqcap \mathsf{R}_2 \end{array} \end{array} \\ \hline \mathsf{R}_1 \sqcap \mathsf{R}_2 \end{array} \tag{6.16}$$

in depth 2. The same transformation can be applied repeatedly to make any general dilemma proof π of a formula F into a bivalent proof π' in depth $2 \cdot D(\pi)$.

The bivalent dilemma proof π' constructed in this way has length $L(\pi') \leq 2^{D(\pi)} \cdot L(\pi)$, so if $D(\pi) = H_{\mathcal{D}}(F)$, it follows by corollary 4.21 on page 58 that $L(\pi') \leq L(\pi)^3$. In other words, given a minimum-depth general dilemma proof π, we can easily find a bivalent proof π' with length (and size) polynomial in $L(\pi)$.

It should be observed, though, that this does not imply that the two proof systems are p-equivalent. The problem is that there might be (unnecessarily) deep but very short general dilemma proofs (say, a proof π in depth $D(\pi) = d$ and length $L(\pi) = \mathrm{O}(d)$), the length of which explode exponentially when the above transformation is applied. The question whether $\mathcal{D}$ and $\mathcal{D}_B$ are p-equivalent or whether $\mathcal{D}$ can be separated from $\mathcal{D}_B$ is open. Note, however, that by (the proof of) theorem 4.35 on page 78, $\mathcal{D}_B$ can be at most quasi-polynomially worse than $\mathcal{D}$.

We summarize the above in a theorem.

Theorem 6.3 (Relation between $\mathcal{D}$ and $\mathcal{D}_B$)
For any propositional logic formula F, the hardness degrees of F in bivalent and general dilemma are within a constant factor, namely:

$$H_{\mathcal{D}}(F) \leq H_{\mathcal{D}_B}(F) \leq 2 \cdot H_{\mathcal{D}}(F).$$

If π is a general dilemma proof for F in minimal depth $H_{\mathcal{D}}(F)$, it is possible to construct in polynomial time a bivalent proof π' with length and size polynomial in the length and size of π. That is, general dilemma restricted to minimum-depth proofs is p-simulated by bivalent dilemma with respect to proof length and size.

If π is any general dilemma proof for F, there is a bivalent proof π' with length and size at most quasi-polynomial in the size of π.

For general dilemma, we know that if a family of formulas F has linearly increasing hardness degree, then the minimum proof length grows exponentially (corollary 4.21). In view of theorem 6.3, this must hold for bivalent dilemma as well. But what about atomic dilemma?

If we inspect corollary 4.21 closer, we see that one cannot simply copy the proof to get a corresponding bound for atomic dilemma. The problem is that the unrolling lemma 4.19 fails. And indeed, the assertion that proof length is exponential in proof hardness is false for atomic dilemma. The formula family CM_n in figure 6.1 has linearly growing atomic hardness degree $H_{\mathcal{D}_A}(CM_n)$ by theorem 6.1. But figure 6.4 gives an atomic NF-dilemma refutation of CM_n in length $\mathrm{O}\left(n^2\right)$, and it is not hard to see that this refutation works in atomic binary dilemma as well. We state this as a theorem.

Theorem 6.4 (Hardness does not imply length for $\mathcal{D}_A$)
There is a family of polynomial-size propositional logic formulas F_n such that $H_{\mathcal{D}_A}(CM_n) = \Omega(n)$ but $L_{\mathcal{D}_A}(F_n \vdash \bot) = n^{\mathrm{O}(1)}$. That is, for atomic dilemma it does not hold that proof length is exponential in proof hardness.

The significance of theorem 6.4 is that it tells us that for atomic dilemma, hardness degree is not a very meaningful concept. Formulas with high atomic dilemma hardness degree are *not* necessarily hard in the sense that they require large proofs. Therefore, our main tool for proving lower bounds on derivation length and size in other dilemma and reductio proof systems will not work for atomic dilemma.

Can we find some other way of separating $\mathcal{D}_B$ from $\mathcal{D}_A$ with respect to proof length and size, or are the two proof systems p-equivalent? It has not been possible to give a definite answer to this question within the framework of this report, but we include some observations below. We first study the question for dilemma restricted to CNF formulas and then turn to general propositional logic formulas.

If we look more closely at the atomic dilemma proof in figure 6.4, we see that it is just a polynomially longer rewritten version of the $\mathcal{RAA}_B$-refutation given in the proof of theorem 6.1 where assumptions over subformulas have been replaced by nested assumptions over the atomic variables in the subformulas. It is not hard to see that an analogous transformation can be applied to the $\mathcal{RAA}_B$-refutation of the CNF formula GT_n outlined in conjecture 6.2 and figure 6.3. In the next proposition, we show how the technique can be generalized further to arbitrary CNF formulas.

$$
\dfrac{
\begin{array}{c}
CM_n \equiv \top \\
\bigvee_{i=1}^{n} R_i^n \equiv \top \\
\bigvee_{j=1}^{n} C_j^n \equiv \top
\end{array}
}{
\dfrac{
\dfrac{
\begin{array}{c}
x_{1,1} \equiv \top \\
C_1^n \equiv \bot
\end{array}
}{
\dfrac{
\begin{array}{c}
x_{1,2} \equiv \top \\
C_2^n \equiv \bot \\
\vdots
\end{array}
}{
\dfrac{
\begin{array}{cc}
x_{1,n-1} \equiv \top & x_{1,n-1} \equiv \bot \\
C_n^{n-1} \equiv \bot & R_1^n \equiv \bot \\
C_n^n \equiv \top & \\
x_{1,n} \equiv \bot & \\
R_1^n \equiv \bot &
\end{array}
}{
\begin{array}{c}
R_1^n \equiv \bot \\
\vdots \\
R_1^n \equiv \bot
\end{array}
}
}
\quad
\begin{array}{c}
x_{1,2} \equiv \bot \\
R_1^n \equiv \bot
\end{array}
}
\quad
\begin{array}{c}
x_{1,1} \equiv \bot \\
R_1^n \equiv \bot
\end{array}
}{
\dfrac{R_1^n \equiv \bot}{R_1^n \equiv \bot}
}
}
$$

$$
\dfrac{
\begin{array}{c}
x_{2,1} \equiv \top \\
\vdots \\
R_2^n \equiv \bot
\end{array}
\qquad
\begin{array}{c}
x_{2,1} \equiv \bot \\
R_2^n \equiv \bot
\end{array}
}{
\begin{array}{c}
R_2^n \equiv \bot \\
\vdots \\
R_n^n \equiv \bot \\
\bigvee_{i=1}^{n} R_i^n \equiv \bot \\
\bot
\end{array}
}
$$

Figure 6.4: Schematic representation of short $\mathcal{D}_A$-refutation of CM_n.

$$
\begin{array}{c}
\mathsf{R} \\
\hline
\begin{array}{cc}
\begin{array}{c}
a_1 \equiv \top \\
\bigvee_{i=1}^{k} a_i \equiv \top \\
\pi_1 \\
\mathsf{R}_1
\end{array}
&
\begin{array}{c}
a_1 \equiv \bot \\
\hline
\begin{array}{cc}
\begin{array}{c}
a_2 \equiv \top \\
\bigvee_{i=1}^{k} a_i \equiv \top \\
\pi_1 \\
\mathsf{R}_1
\end{array}
&
\begin{array}{c}
a_2 \equiv \bot \\
\vdots \\
\hline
\begin{array}{cc}
\begin{array}{c}
a_k \equiv \top \\
\bigvee_{i=1}^{k} a_i \equiv \top \\
\pi_1 \\
\mathsf{R}_1
\end{array}
&
\begin{array}{c}
a_k \equiv \bot \\
\bigvee_{i=1}^{k} a_i \equiv \bot \\
\pi_2 \\
\mathsf{R}_2
\end{array}
\end{array} \\
\hline
\mathsf{R}_1 \sqcap \mathsf{R}_2 \\
\vdots \\
\mathsf{R}_1 \sqcap \mathsf{R}_2
\end{array}
\end{array} \\
\hline
\mathsf{R}_1 \sqcap \mathsf{R}_2
\end{array}
\end{array} \\
\hline
\mathsf{R}_1 \sqcap \mathsf{R}_2
\end{array}
$$

Figure 6.5: Atomic NF-dilemma derivation in the proof of proposition 6.5.

Proposition 6.5
Suppose that F_n is a family of contradictory CNF formulas with bivalent NF-dilemma refutations $\pi_n : {F_n}^{\top} \Rightarrow \bot_{F_n{}^+}$. Then the length of refuting F_n in atomic NF-dilemma is

$$L_{\mathcal{D}_A}(F_n \vdash \bot) \le W(F_n)^{D(\pi_n)} \cdot L(\pi_n).$$

In particular, if the formulas F_n have width $W(F_n) = \mathrm{O}(n)$ and minimal refutation length $L_{\mathcal{D}_B}(F_n \vdash \bot) = \Omega(n)$, and if the bivalent NF-dilemma refutations π_n have depth $D(\pi_n) = \mathrm{O}(1)$, then the length of refuting F_n in atomic NF-dilemma $L_{\mathcal{D}_A}(F_n \vdash \bot)$ is polynomial in $L_{\mathcal{D}_B}(F_n \vdash \bot)$.

Proof: Suppose that $\bigvee_{i=1}^{k} a_i$ is a clause in F_n and that

$$
\pi = \begin{array}{c}
\mathsf{R} \\
\hline
\begin{array}{cc}
\bigvee_{i=1}^{k} a_i \equiv \top & \bigvee_{i=1}^{k} a_i \equiv \bot \\
\pi_1 & \pi_2 \\
\mathsf{R}_1 & \mathsf{R}_2
\end{array} \\
\hline
\mathsf{R}_1 \sqcap \mathsf{R}_2
\end{array}
\tag{6.17}
$$

is a bivalent NF-dilemma (sub)derivation in depth 1. Then figure 6.5 presents an equivalent atomic derivation in length at most $k \cdot L(\pi)$. The transformation from (6.17) to the derivation in figure 6.5 can be applied in a bottom-up fashion to π_n to yield an atomic refutation of F_n in length at most $W(F_n)^{D(\pi_n)} \cdot L(\pi_n)$.

This proves the first part of the proposition. The second part is an easy consequence. □

It follows from proposition 6.5 that if we are looking for CNF formulas to separate bivalent NF-dilemma from atomic NF-dilemma, the only candidates are formula families F_n with $H_{\mathcal{D}_B}(F_n) = \omega(1)$ and/or width $W(F_n) = \omega(n)$. Note that these requirements disqualify for example many CNF formula families studied in the context of resolution. Of course, this does not answer the question whether in fact there are CNF formulas that separate bivalent and atomic dilemma, but it tells us that at least we cannot expect to find any simple examples of such separations.

For general propositional logic formulas the matter seems more clear-cut. Consider the following separation suggested by [51].

Fix a formula F^* with $H_{\mathcal{D}_B}(F^*) = h \geq 1$ and $\mathit{Vars}(F^*) = \{x_1, \ldots, x_k\}$. Pick a family of polynomial-size formulas G_n with bivalent dilemma hardness $H_{\mathcal{D}_B}(G_n) = \Omega(n)$ and let $G_n^{(j)}$ denote the formula G_n with variables tagged so that $\mathit{Vars}(G_n^{(j)}) \cap \mathit{Vars}(G_n^{(j')}) = \emptyset$ if $j \neq j'$.

Now define a new formula family $\{F_n\}_{n=1}^{\infty}$ by

$$F_i := F^*[G_i^{(1)}/x_1, \ldots, G_i^{(k)}/x_k] \tag{6.18}$$

(i.e. each x_j is substituted by a unique tagged version of G_i). Obviously, for all F_n the bivalent dilemma hardness is $H_{\mathcal{D}_B}(F_n) = H_{\mathcal{D}_B}(F^*) = h$ and the same refutation as for F^* can be used to refute F_n in constant length and polynomial size (since we can ignore the new subformulas and make assumptions only over the subformulas in F^*). For atomic dilemma, however, we claim that the refutation length grows exponentially with n. Intuitively speaking, the reason for this is that in order to derive consequences about F_n from atomic assumptions, we must first descend into the subformulas and derive consequences in the subdomains $\mathit{Sub}\left(G_i^{(j)}\right)$, and this requires exponential length even for bivalent dilemma.

Needless to say, the above argument needs to be formalized. We have not done this out of practical considerations, and therefore leave this result as a conjecture.

Conjecture 6.6 (Exponential Separation of $\mathcal{D}_B$ from $\mathcal{D}_A$)
There exists a family of polynomial-size propositional logic formulas F_n which separates bivalent dilemma from atomic dilemma exponentially with respect to proof length and size.

6.2 Dilemma vs. Reductio

We now broaden our perspective and include the atomic, bivalent and general reductio proof systems $\mathcal{RAA}_A$, $\mathcal{RAA}_B$ and $\mathcal{RAA}$ in our comparative study.

It is trivially true that $\mathcal{D}_A$ simulates $\mathcal{RAA}_A$, $\mathcal{D}_B$ simulates $\mathcal{RAA}_B$ and $\mathcal{D}$ simulates $\mathcal{RAA}$ with respect to any measure M since any RAA derivation is a dilemma derivation. Just as trivially, $\mathcal{RAA}_B$ simulates $\mathcal{RAA}_A$ and $\mathcal{RAA}$ simulates $\mathcal{RAA}_B$ with respect to any measure M.

In this section we establish some results in the opposite direction. Our main focus is the comparison of dilemma and reductio systems with the same restrictions on dilemma rule assumptions. The effectiveness of the dilemma proof system is due to (at least) three factors: the large set of propagation rules, the dilemma rule combining branching and merging and the possibility to branch over the truth or falsehood of arbitrary subformulas of the formula to be proved. Juxtaposing $\mathcal{D}_A$, $\mathcal{D}_B$ and $\mathcal{D}$ (as was done in section 6.1) tells us how important it is to be able to branch over arbitrary subformulas. Comparing atomic, bivalent and general dilemma with corresponding reductio systems reveals how much of the proof power is lost by removing merging, and thus in a sense makes it possible to weigh the relative importance of the last two factors above.

For atomic dilemma the merging part of the dilemma rule is crucial, as the next theorem shows.

Theorem 6.7 (Exponential separation of $\mathcal{D}_A$ from $\mathcal{RAA}_A$)
There is a family of polynomial-size formulas F_n such that $L_{\mathcal{D}_A}(F_n \vdash \perp) = n^{\mathrm{O}(1)}$ but $L_{\mathcal{RAA}_A}(F_n \vdash \perp) = \exp\bigl(\Omega(n)\bigr)$. That is, atomic dilemma is exponentially stronger than atomic reductio with respect to proof length (and proof size).

Proof: Consider the formulas CM_n defined in figure 6.1 on page 94. By theorem 6.1, $H_{\mathcal{D}_A}(CM_n) = \Omega(n)$, so certainly $H_{\mathcal{RAA}_A}(CM_n) = \Omega(n)$ and by theorem 5.10 we have $L_{\mathcal{RAA}_A}(F_n \vdash \perp) = \exp\bigl(\Omega(n)\bigr)$. For atomic dilemma, however, figure 6.4 on page 99 gives a polynomial-length $\mathcal{D}_A$-refutation of CM_n. The theorem follows. □

Moreover, noting that the proof of theorem 6.1 in fact presents a polynomial-length bivalent *reductio* refutation of CM_n in depth 1, we have the following two corollaries.

Corollary 6.8 ($\mathcal{RAA}_B$ is exponentially stronger than $\mathcal{RAA}_A$)
Bivalent reductio is linearly stronger than atomic reductio with respect to proof hardness and exponentially stronger with respect to proof length and size.

Corollary 6.9 (Separation of $\mathcal{RAA}_B$ from $\mathcal{D}_A$ w.r.t. hardness)
There is a family of polynomial-size propositional logic formulas F_n which separates bivalent reductio from atomic dilemma linearly with respect to hardness degree.

Remark 6.10 Note that if conjecture 6.2 is true, the results in theorem 6.7 and corollaries 6.8 and 6.9 hold also for the more restricted class of CNF formulas.

As can be seen from theorem 6.4 and figure 6.4, the separation with respect to hardness in corollary 6.9 does not yield any separation with respect to the in this context more interesting measures of proof length and size. It is not known whether there are such separations of $\mathcal{RAA}_B$ from $\mathcal{D}_A$. We have not studied this question in any detail, but the answer is likely to be the same as that in proposition 6.5 and conjecture 6.6 concerning $\mathcal{D}_B$ and $\mathcal{D}_A$. In the opposite direction, since $\mathcal{D}_A$ can merge branches without having to derive a contradiction and $\mathcal{RAA}_B$ cannot, it might very well be the case that there are formula families which separate $\mathcal{D}_A$ from $\mathcal{RAA}_B$ with respect to proof hardness, length or size.

For $\mathcal{D}_B$ and $\mathcal{RAA}_B$, all we have is a logarithmic separation with respect to hardness. This separation is not our result but is due to [53]. Since it has not been published, we state it here for reference.

Theorem 6.11 (Separation of $\mathcal{D}_B$ from $\mathcal{RAA}_B$ w.r.t. hardness [53])
There is a family of polynomial-size propositional logic formulas F_n which separates bivalent dilemma from bivalent reductio logarithmically with respect to hardness degree (i.e. $H_{\mathcal{D}_B}(F_n) = \mathrm{O}(1)$ but $H_{\mathcal{RAA}_B}(F_n) = \Omega(\log n)$).

The formulas which provide this separation are the so called *Urquhart formulas* U_n, defined by

$$U_n := \Bigl(\bigl(\bigl(\bigl(\ldots\bigl((x_1 \leftrightarrow x_2) \leftrightarrow x_3\bigr) \leftrightarrow \ldots\bigr) \leftrightarrow x_n\bigr) \leftrightarrow x_1\bigr) \leftrightarrow \ldots\Bigr) \leftrightarrow x_n \qquad (6.19)$$

(the formula in example 4.3 on page 66 is the Urquhart formula U_2). According to [53], $H_{\mathcal{D}_B}(U_1) = 0$, $H_{\mathcal{D}_B}(U_2) = 1$ and $H_{\mathcal{D}_B}(U_n) = 2$ for $n > 2$, but in $\mathcal{RAA}_B$ the hardness grows at least logarithmically with n. However, [33] shows that the growth is *at most* logarithmic, and that there are bivalent RAA proofs of length and size polynomial in n. Thus, theorem 6.11 yields no superpolynomial separation of $\mathcal{D}_B$ from $\mathcal{RAA}_B$ with respect to proof length and size.

It is an open question whether $\mathcal{D}_B$ and $\mathcal{RAA}_B$ can be separated superpolynomially with respect to to proof length and size or superlogarithmically with respect to proof hardness (from which superpolynomial separation of length and size would follow).

Finally, we consider the general proof systems $\mathcal{D}$ and $\mathcal{RAA}$. Obviously, any general dilemma derivation

$$\begin{array}{c} \mathsf{R} \\ \hline \begin{array}{cc} \mathsf{R}[\psi] & \mathsf{R}[\psi^C] \\ \pi_1 & \pi_2 \\ \mathsf{R}_1[\phi] & \mathsf{R}_2[\phi] \end{array} \\ \hline \mathsf{R}[\phi] \end{array} \tag{6.20}$$

of depth 1 (where ψ and ϕ are single associations) can be transformed to an equivalent general RAA derivation

$$\begin{array}{c} \mathsf{R} \\ \hline \begin{array}{cc} \mathsf{R}[\phi] & \begin{array}{c} \mathsf{R}[\phi^C] \\ \hline \begin{array}{cc} \mathsf{R}[\phi^C, \psi] & \begin{array}{c} \mathsf{R}[\phi^C, \psi^C] \\ \pi_2 \\ \perp_{\mathsf{R}} \end{array} \end{array} \\ \hline \mathsf{R}[\phi^C, \psi] \\ \pi_1 \\ \perp_{\mathsf{R}} \end{array} \end{array} \\ \hline \mathsf{R}[\phi] \end{array} \tag{6.21}$$

of depth 2. It is easy to generalize the rewriting of (6.20) to (6.21) inductively to a transformation of any dilemma proof π to an general RAA proof in depth at most $2 \cdot D(\pi)$, which proves the first part of the next theorem.

Theorem 6.12 (Relation between $\mathcal{D}$ and $\mathcal{RAA}$)
For any propositional logic formula F, the hardness degrees of F in general dilemma $\mathcal{D}$ and reductio $\mathcal{RAA}$ are within a constant factor, namely:

$$H_{\mathcal{D}}(F) \leq H_{\mathcal{RAA}}(F) \leq 2 \cdot H_{\mathcal{D}}(F).$$

As to proof size, $\mathcal{RAA}$ is at most quasi-polynomially worse than $\mathcal{D}$.

Proof: The first part was proved above. For the proof size, by (4.20) it is sufficient to prove that $L_{\mathcal{RAA}}(\vdash F)$ is quasi-polynomial in $S_{\mathcal{D}}(\vdash F)$.

For a 0-easy formula F all 0-depth proofs are polynomial in $S(F) \leq S_{\mathcal{D}}(\vdash F)$, so suppose $H_{\mathcal{D}}(F) = h \geq 1$. Then $H_{\mathcal{RAA}}(F) \leq 2h$ and it follows from the proof of theorem 4.23 that

$$L_{\mathcal{RAA}}(\vdash F) \leq \mathrm{O}\left(S(F)^{2h+1}\right). \tag{6.22}$$

Also, we have

$$S_{\mathcal{D}}(\vdash F) \geq L_{\mathcal{D}}(\vdash F) \geq 2^{h/2} \tag{6.23}$$

by corollary 4.21. Using (6.22) and (6.23) we get

$$L_{\mathcal{RAA}}(\vdash F) \leq \left(2^{h/2}\right)^{6\log S(F)} \leq \left(S_{\mathcal{D}}(\vdash F)\right)^{6\log S_{\mathcal{D}}(\vdash F)}, \tag{6.24}$$

which proves the theorem. □

There are no known separations between $\mathcal{D}$ and $\mathcal{RAA}$, and even if it seems likely that $\mathcal{D}$ should be strictly stronger than $\mathcal{RAA}$, theorem 6.12 shows that it cannot be exponentially stronger.

As a by-product of our comparisons between dilemma an reductio, we obtained a full characterization of the relation between $\mathcal{RAA}_B$ and $\mathcal{RAA}_A$ with respect to all three measures of hardness, length and size (corollary 6.8). In contrast, we have no nontrivial results concerning the relation between $\mathcal{RAA}$ and $\mathcal{RAA}_B$. Our only result in this context is an immediate consequence of theorems 6.3, 6.11 and 6.12.

Corollary 6.13 (Separation of $\mathcal{RAA}$ from $\mathcal{RAA}_B$ w.r.t. hardness) *There is a family of polynomial-size propositional logic formulas F_n which separates general reductio from bivalent reductio logarithmically with respect to hardness degree.*

6.3 Dilemma vs. Tree Resolution

Before even getting around to describing dilemma and reductio, we spent the better part of two chapters of the thesis discussing resolution. The reason why resolution is interesting is that it is the basis of many "real-life" proof search algorithms. Comparing resolution and the dilemma proof system underlying Stålmarck's method can give us theoretical bounds on the performance of proof methods in the two proof systems. Although such bounds more often than not are of little immediate practical significance, they can still provide interesting insights as to why we can expect certain proof methods to be effective or not on certain types of problems.[2]

In the next section, we study general resolution. Before that, as a warm-up we prove some easy results on dilemma/reductio and tree-like resolution $\mathcal{T}$ (which is the proof system corresponding to the DLL proof search algorithm in example 2.5 on page 24).

We start by proving that atomic NF-dilemma and bivalent reductio can be separated from tree-like resolution exponentially with respect to of proof length and size.

Theorem 6.14 (Exponential separation of $\mathcal{D}_A$ and $\mathcal{RAA}_B$ from $\mathcal{T}$) *There exists a family of polynomial-size contradictory CNF formulas F_n such that $L_{\mathcal{D}_A}(F_n \vdash \bot) = n^{\mathrm{O}(1)}$ and $L_{\mathcal{RAA}_B}(F_n \vdash \bot) = n^{\mathrm{O}(1)}$ but $L_{\mathcal{T}}(F_n \vdash \bot) = \exp\left(\Omega\left(n/\log n\right)\right)$.*

[2] In fact, as it happened our study of dilemma proof systems for CNF coincided with tests at Prover Technology AB of Stålmarck's method on CNF formulas, and in this case there was an interesting agreement between theoretically founded intuition and practical experience.

Proof: Consider the pebbling contradictions Peb_G (definition 3.21 in section 3.2). By theorem 3.18, the length of a minimal tree-like resolution refutation grows exponentially, but it it easy to transform the resolution refutation described in the proof of lemma 3.22 to an atomic dilemma refutation or bivalent RAA refutation in constant depth and length $\mathrm{O}(S(Peb_G))$. $\square$

If we remove the possibility to branch over subformulas in bivalent reductio or the possibility to merge the branches in atomic dilemma, we get a proof system which is polynomially equivalent to tree-like resolution.

Theorem 6.15 ($\mathcal{RAA}_A$ and $\mathcal{T}$ are p-equivalent w.r.t. length)
Atomic NF-reductio and tree-like resolution are polynomially equivalent with respect to proof length.

Remark 6.16 Because of a technical snag, we do not get p-equivalence for the proof sizes. Any formula relation based dilemma or RAA derivation π_D from a formula F must have size $S(\pi_D) \geq S(F)$ since by definition π_D consists of a sequence of formula relations and a formula relation on F is at least as large as F. Now suppose that we define a formula family

$$F_n := x \wedge \overline{x} \wedge \bigvee_{1 \leq i \leq n} y_i. \tag{6.25}$$

The size of $\mathcal{RAA}_A$-refutations of F_n grows at least linearly in n, although F_n can be refuted in $\mathcal{RAA}_A$ (and $\mathcal{T}$) by deriving a contradiction from x and $\overline{x}$ in length 3. It follows that there are formula families with $\mathcal{RAA}_A$- and $\mathcal{T}$-refutations in constant length but $\mathcal{RAA}_A$-proof size more than exponential in $\mathcal{T}$-proof size.

One way of eliminating this rather artificial separation of $\mathcal{T}$ from $\mathcal{RAA}_A$ would be to require that all tree resolution refutations π_T start by listing all clauses in the formula F to be refuted. A more elegant way of solving the problem would be to modify the definitions in chapter 4 so that derivations are defined in terms of formula relations R with dynamic domains $Sub(\mathsf{R})$, and so that these domains consist of (in most cases strict) subsets of triplets of F (definition 4.27). To simplify the exposition in this thesis, however, we have chosen to define derivations on static formula relation domains in accordance with the conventional description of the dilemma proof system and Stålmarck's method in [49].

Sketch of proof [of theorem 6.15]: If we consider a tree resolution refutation π_T as a binary tree (turned upside down) with

- each leaf labelled by the clause (or one of the clauses) in the CNF formula falsified at the leaf,
- each internal node labelled by the clause derived at that node,
- each edge labelled by the literal eliminated by the resolution rule when moving from the children to the parent node,

then it is straightforward to construct an (equivalence-based) $\mathcal{RAA}_A$-refutation in length $L(\pi_T)$ which branches over the variables mentioned at the edges and derives contradictions in the branches by falsifying the clauses mentioned in corresponding leaves. We leave the details to the reader.

The other direction is somewhat more complicated. Intuitively, dilemma with only atomic branching and without merging is so weak for CNF formulas that all it can do is essentially to assume variables true or false and apply unit propagation. But this is what tree-like resolution does, too, so the two systems are p-equivalent.

More formally, let π_D be an atomic NF-reductio refutation of a CNF formula F. Suppose without loss of generality that π_D starts by deriving $C \equiv \top$ in depth 0 for all clauses $C \in F$ used in the proof. After this first subderivation, no rules for conjunction (figure 5.1) need be used.

We claim that the following invariants hold for all formula relations R in π_D:

1. if $\bigvee_{i=1}^{k} a_i \equiv \neg \bigvee_{j=1}^{l} b_i$ in a formula relation R in π_D, then $\bigvee_{i=1}^{k} a_i$ and $\bigvee_{j=1}^{l} b_i$ are members of determinate equivalence classes in R (i.e. the *TRUE*- and *FALSE*-classes),

2. if $\bigvee_{i=1}^{k} a_i \equiv \bot$ in R, then $a_i \equiv \bot$ already holds for all $i = 1, \ldots, k$ in R.

The invariants are certainly true at the beginning of the derivation. Consider the propagation rules for disjunction in figure 5.2 on page 86. Because of the invariants, the only rules that can be used in π_D are (D4_{NF}), (D5_{NF}) and (D6_{NF}), and these rules preserve the invariants, as do branching and merging in atomic reductio.

It follows that contradictions in the branches can only be derived by proving $\bigvee_{i=1}^{k} a_i \equiv \top$ and $\bigvee_{i=1}^{k} a_i \equiv \bot$ for some (sub)clause of F and never by deriving $D \equiv \neg D$ in some indeterminate equivalence class. But if $\bigvee_{i=1}^{k} a_i \equiv \bot$, then we already have $a_i \equiv \bot$ for all $i = 1, \ldots, k$, so without loss of generality we can assume that all derived contradictions are of the type $a_i \equiv \top$ and $a_i \equiv \bot$ for some literal a_i in F.

Now one can show that a DLL procedure can keep track of the set of true clauses in every step in π_D in the sense that if D is a clause (possibly a literal) in the *TRUE*-class, then the DLL procedure has already proven a clause $D' \subseteq D$ true (or assumed it true, in the case of a literal). The DLL procedure can ignore all applications of rules (D4_{NF}) and (D6_{NF}) (which can be considered as applications of weakening). For all applications of the rule (D5_{NF}) which derive a new true clause, by induction the DLL procedure can derive the same clause (or a subclause) true in a polynomial number of steps. When π_D branches over some variable, so does the DLL procedure. When π_D arrives at a contradiction, so does the DLL procedure (if not before). It follows that the tree resolution refutation produced by the DLL procedure is polynomial in the length of π_D. Again, we leave the details to the reader. □

Remark 6.17 (Hardness in tree-like resolution) The definitions of proof depth and hardness in chapter 4 can be generalized to any tree-like proof system $\mathcal{P}$ as follows. Suppose that π is a proof in $\mathcal{P}$ represented as a binary tree. Define the *proof depth* of leafs v in π as $D(v) := 0$. For inner nodes v with children v_1 and v_2, define $D(v) := \max\{D(v_1), D(v_2)\}$ if $D(v_1) \neq D(v_2)$ and $D(v) := D(v_1) + 1$ otherwise. Now the depth of the proof π is defined as the proof depth of the root of the tree, and the hardness of a formula F

in $\mathcal{P}$ is defined as $H_{\mathcal{P}}(F) := \min_\pi \{D(\pi)\}$, where the minimum is taken over all $\mathcal{P}$-proofs π for F.

If we define proof depth and hardness in tree resolution as described above, it follows immediately from the proof of theorem 6.15 that $H_{\mathcal{T}}(F) \geq H_{\mathcal{RAA}_A}(F)$ for all unsatisfiable CNF formulas F. We have chosen not to study the hardness measure in tree resolution (or its possible extension to general resolution), however, and instead confine ourselves to noting that such a possibility exist.

From theorems 6.14 and 6.15 we get the following corollary.

Corollary 6.18 ($\mathcal{D}_A$ and $\mathcal{RAA}_B$ are exponentially stronger than $\mathcal{T}$)
Atomic NF-dilemma and bivalent NF-reductio are both exponentially stronger than tree resolution with respect to proof length.

The interesting thing is that since the the pebbling contradictions Peb_G are 1-easy not only for normal form but also for *binary* $\mathcal{D}_A$ (and $\mathcal{RAA}_B$), this shows that there are formulas F where Stålmarck's method will find a proof in time polynomial in the size $S(F)$, but where *any* DLL procedure, no matter what splitting rule it uses, must take exponential time.

The reader might have noticed that we did not separate atomic dilemma from atomic RAA with respect to hardness in section 6.2. Although this is a purely academic question in view of the exponential separations in theorem 6.7, we note that the results in this section provide the expected linear separation of $\mathcal{D}_A$ from $\mathcal{RAA}_A$ in terms of proof hardness.

Corollary 6.19 ($\mathcal{D}_A$ is linearly stronger than $\mathcal{RAA}_A$ w.r.t. hardness)
There is a family of polynomial-size propositional logic formulas F_n such that $H_{\mathcal{D}_A}(F_n) = \mathrm{O}(1)$ but $H_{\mathcal{RAA}_A}(F_n) = \Omega(n)$.

This is an immediate consequence of theorem 6.14 (which holds also for parenthesized "propositional logic versions" of the pebbling contradictions) and theorem 6.15 combined with theorem 5.7 (and remark 5.3).

6.4 Dilemma vs. General Resolution

We conclude this chapter by presenting our results relating dilemma to general resolution. As before, resolution considered as a proof system is denoted $\mathcal{R}$.

Our most important result, from which all other results in this section follow, is that for CNF formulas with maximum clause width k, normal form dilemma proofs in depth d and length L can be translated to resolution proofs in width $\mathrm{O}(kd)$ and length $\left(Lk^d\right)^{\mathrm{O}(1)}$.

Theorem 6.20 (Depth-width relation of dilemma and resolution)
Suppose that F is an unsatisfiable CNF formula in width $W(F) = k$. Then any equivalence-based general NF-dilemma refutation π_D of F in depth $D(\pi_D) = d$ and length $L(\pi_D) = L$ can be translated to a resolution refutation π_R of F in width

$$W(\pi_R) \leq (2d+3)(k-1)$$

and length

$$L(\pi_R) \leq 2L \cdot k^{2d+3}.$$

We describe the ideas behind the proof below. The detailed proof is rather tedious and is therefore deferred to section B.2.

Sketch of proof: Let F be an unsatisfiable CNF formula with maximum clause width k and suppose that π_D is an equivalence-based general NF-dilemma refutation of F. We assume (without loss of generality) that π_D starts by deriving $C \equiv \top$ for all clauses $C \in F$ used in the proof.

Every line in the proof π_D consists of some equivalence ϕ derived under (open) assumptions $\psi_1, \ldots, \psi_i$ from preceding lines in the proof (within the same scope as ψ, see definition A.2 on page 132). If we denote this

$$\psi_1 \Rightarrow \ldots \Rightarrow \psi_i \Rightarrow \phi, \tag{6.26}$$

we can transform an equivalence-based dilemma proof to a *line-based* dilemma proof by replacing the derivation steps by a sequence of lines on the form (6.26). Note that all equivalences $\psi_1, \ldots, \psi_i, \phi$ are over (sub)clauses of F (except for the first line $F \equiv \top$).

In the next step, we translate each line in the line-based proof to (at most) k^{2i+2} clauses by interpreting (6.26) as

$$\neg\psi_1 \vee \ldots \vee \neg\psi_i \vee \phi \tag{6.27}$$

and rewriting this formula to to a set of CNF clauses. Let us denote this translation by $\mathit{CNF}\,(\psi_1 \Rightarrow \ldots \Rightarrow \psi_i \Rightarrow \phi)$. Perhaps it is easiest explained by an example: the line-based dilemma derivation line

$$x_1 \vee x_2 \equiv y_1 \vee y_2 \Rightarrow z_1 \vee z_2 \equiv w_1 \vee w_2 \tag{6.28}$$

(i.e. $z_1 \vee z_2 \equiv w_1 \vee w_2$ derived under the assumption $x_1 \vee x_2 \equiv y_1 \vee y_2$) is rewritten to the set of CNF clauses in figure 6.6.

Finally, we prove by induction that for each line $\psi_1 \Rightarrow \ldots \Rightarrow \psi_i \Rightarrow \phi$ in the line-based dilemma proof, the clauses in $\mathit{CNF}\,(\psi_1 \Rightarrow \ldots \Rightarrow \psi_i \Rightarrow \phi)$ can be derived in resolution from sets of clauses corresponding to preceding lines in the dilemma proof in length at most $2k \cdot L\,(\mathit{CNF}\,(\psi_1 \Rightarrow \ldots \Rightarrow \psi_i \Rightarrow \phi))$ and width at most $(2d' + 3)\,(k-1)$, where d' is the maximum depth in the derivation so far, $(2d' + 2)\,(k-1)$ is the maximum width of the CNF clause translations so far and $k-1$ is the maximum number of extra literals needed during the resolution derivation of $\mathit{CNF}\,(\psi_1 \Rightarrow \ldots \Rightarrow \psi_i \Rightarrow \phi)$. It follows that if

$$\begin{gathered} \psi_{1,1} \Rightarrow \ldots \Rightarrow \psi_{1,i_1} \Rightarrow \phi_1 \\ \vdots \\ \psi_{n,1} \Rightarrow \ldots \Rightarrow \psi_{n,i_n} \Rightarrow \phi_n \end{gathered} \tag{6.29}$$

is a dilemma derivation in length $L\,(\pi_D) = L$ and depth $D\,(\pi_D) = d$, then

$$\begin{gathered} \mathit{CNF}\,(\psi_{1,1} \Rightarrow \ldots \Rightarrow \psi_{1,i_1} \Rightarrow \phi_1) \\ \vdots \\ \mathit{CNF}\,(\psi_{n,1} \Rightarrow \ldots \Rightarrow \psi_{n,i_n} \Rightarrow \phi_n) \end{gathered} \tag{6.30}$$

is the backbone of a corresponding resolution derivation, the gaps of which can be completed in length at most $2L \cdot k^{2d+3}$ and width at most $(2d+3)\,(k-1)$.

$$
\begin{aligned}
\{&x_1 \lor x_2 \lor y_1 \lor y_2 \lor \overline{z}_1 \lor w_1 \lor w_2,\\
&x_1 \lor x_2 \lor y_1 \lor y_2 \lor \overline{z}_2 \lor w_1 \lor w_2,\\
&x_1 \lor x_2 \lor y_1 \lor y_2 \lor z_1 \lor z_2 \lor \overline{w}_1,\\
&x_1 \lor x_2 \lor y_1 \lor y_2 \lor z_1 \lor z_2 \lor \overline{w}_2,\\
&\overline{x}_1 \lor \overline{y}_1 \lor \overline{z}_1 \lor w_1 \lor w_2,\\
&\overline{x}_1 \lor \overline{y}_1 \lor \overline{z}_2 \lor w_1 \lor w_2,\\
&\overline{x}_1 \lor \overline{y}_1 \lor z_1 \lor z_2 \lor \overline{w}_1,\\
&\overline{x}_1 \lor \overline{y}_1 \lor z_1 \lor z_2 \lor \overline{w}_2,\\
&\overline{x}_1 \lor \overline{y}_2 \lor \overline{z}_1 \lor w_1 \lor w_2,\\
&\overline{x}_1 \lor \overline{y}_2 \lor \overline{z}_2 \lor w_1 \lor w_2,\\
&\overline{x}_1 \lor \overline{y}_2 \lor z_1 \lor z_2 \lor \overline{w}_1,\\
&\overline{x}_1 \lor \overline{y}_2 \lor z_1 \lor z_2 \lor \overline{w}_2,\\
&\overline{x}_2 \lor \overline{y}_1 \lor \overline{z}_1 \lor w_1 \lor w_2,\\
&\overline{x}_2 \lor \overline{y}_1 \lor \overline{z}_2 \lor w_1 \lor w_2,\\
&\overline{x}_2 \lor \overline{y}_1 \lor z_1 \lor z_2 \lor \overline{w}_1,\\
&\overline{x}_2 \lor \overline{y}_1 \lor z_1 \lor z_2 \lor \overline{w}_2,\\
&\overline{x}_2 \lor \overline{y}_2 \lor \overline{z}_1 \lor w_1 \lor w_2,\\
&\overline{x}_2 \lor \overline{y}_2 \lor \overline{z}_2 \lor w_1 \lor w_2,\\
&\overline{x}_2 \lor \overline{y}_2 \lor z_1 \lor z_2 \lor \overline{w}_1,\\
&\overline{x}_2 \lor \overline{y}_2 \lor z_1 \lor z_2 \lor \overline{w}_2\}
\end{aligned}
$$

Figure 6.6: Translation of $x_1 \lor x_2 \equiv y_1 \lor y_2 \Rightarrow z_1 \lor z_2 \equiv w_1 \lor w_2$ to CNF clauses.

For a simple example of the transformation from equivalence-based dilemma via line-based dilemma to resolution, see the refutations in figures 6.7 and 6.8 of the formula GT_3 (defined in figure 6.2 on page 95). For clarity, we have filled in the gaps in the "backbone" resolution proof in figure 6.8 with parenthesized clauses and indicated from which previous lines in the proof these clauses can be derived. We let figures 6.7 and 6.8 serve as a "proof by example" and refer the interested reader to the formal proof in section B.2 for the details. □

Fix the clause width k, and suppose that π_D is a dilemma refutation of a k-CNF formula F in minimal depth $H(F)$. Then the translation to resolution in the proof of theorem 6.20 yields a resolution refutation π_R at most polynomially larger than π_D. That is, for k-CNF formulas dilemma restricted to minimum-depth proofs is p-simulated by resolution. We state this as a theorem.

Theorem 6.21 (Resolution p-simulates minimum-depth dilemma) *Let $k \geq 2$ be a fix integer and let F be an unsatisfiable k-CNF formula. Suppose that π_D is a general NF-dilemma refutation of F in minimal depth $H(F)$.*

Then it is possible to construct in polynomial time a resolution refutation π_R with size $S(\pi_R)$ polynomial in $S(\pi_D)$. That is, for k-CNF formulas resolution p-simulates dilemma restricted to minimum-depth proofs.

$$\dfrac{\begin{array}{c} GT_3 \equiv \top \\ A(1,3,2) \equiv \top \\ A(2,3,1) \equiv \top \\ B(1,2) \equiv \top \\ B(1,3) \equiv \top \\ B(2,3) \equiv \top \\ C_3(1) \equiv \top \\ C_3(2) \equiv \top \\ C_3(3) \equiv \top \end{array}}{\dfrac{\begin{array}[t]{c} x_{1,2} \equiv \top \\ x_{2,1} \equiv \bot \\ x_{3,1} \equiv \top \\ x_{2,3} \equiv \bot \\ x_{1,3} \equiv \bot \\ C_3(3) \equiv \bot \\ \bot \end{array} \qquad\qquad \begin{array}[t]{c} x_{1,2} \equiv \bot \\ x_{3,2} \equiv \top \\ x_{1,3} \equiv \bot \\ x_{2,3} \equiv \bot \\ C_3(3) \equiv \bot \\ \bot \end{array}}{\bot}}$$

Figure 6.7: Equivalence-based dilemma refutation of GT_3.

$$\begin{array}{lll}
1. & \overline{x}_{1,3} \vee \overline{x}_{3,2} \vee x_{1,2} & [\ \mathit{CNF}\ (A(1,3,2) \equiv \top)\ ] \\
2. & \overline{x}_{2,3} \vee \overline{x}_{3,1} \vee x_{2,1} & [\ \mathit{CNF}\ (A(2,3,1) \equiv \top)\ ] \\
3. & \overline{x}_{1,2} \vee \overline{x}_{2,1} & [\ \mathit{CNF}\ (B(1,2) \equiv \top)\ ] \\
4. & \overline{x}_{1,3} \vee \overline{x}_{3,1} & [\ \mathit{CNF}\ (B(1,3) \equiv \top)\ ] \\
5. & \overline{x}_{2,3} \vee \overline{x}_{3,2} & [\ \mathit{CNF}\ (B(2,3) \equiv \top)\ ] \\
6. & x_{2,1} \vee x_{3,1} & [\ \mathit{CNF}\ (C_3(1) \equiv \top)\ ] \\
7. & x_{1,2} \vee x_{3,2} & [\ \mathit{CNF}\ (C_3(2) \equiv \top)\ ] \\
8. & x_{1,3} \vee x_{2,3} & [\ \mathit{CNF}\ (C_3(3) \equiv \top)\ ] \\
9. & \overline{x}_{1,2} \vee \overline{x}_{2,1} & [\ \mathit{CNF}\ (x_{1,2} \equiv \top \Rightarrow x_{2,1} \equiv \bot)\ ] \\
10. & \overline{x}_{1,2} \vee x_{3,1} & [\ \mathit{CNF}\ (x_{1,2} \equiv \top \Rightarrow x_{3,1} \equiv \top)\ ] \\
11. & (\overline{x}_{1,2} \vee \overline{x}_{2,3} \vee \overline{x}_{3,1}) & [\ \text{By resolution of lines 2 and 3}\ ] \\
12. & \overline{x}_{1,2} \vee \overline{x}_{2,3} & [\ \mathit{CNF}\ (x_{1,2} \equiv \top \Rightarrow x_{2,3} \equiv \bot)\ ] \\
13. & \overline{x}_{1,2} \vee \overline{x}_{1,3} & [\ \mathit{CNF}\ (x_{1,2} \equiv \top \Rightarrow x_{1,3} \equiv \bot)\ ] \\
\begin{array}{l} 14. \\ 15. \end{array} & \left.\begin{array}{l} \overline{x}_{1,2} \vee \overline{x}_{1,3} \\ \overline{x}_{1,2} \vee \overline{x}_{2,3} \end{array}\right\} & [\ \mathit{CNF}\ (x_{1,2} \equiv \top \Rightarrow C_3(3) \equiv \bot)\ ] \\
16. & (\overline{x}_{1,2} \vee x_{2,3}) & [\ \text{By resolution of lines 8 and 14}\ ] \\
17. & \overline{x}_{1,2} & [\ \mathit{CNF}\ (x_{1,2} \equiv \top \Rightarrow \bot)\ ] \\
18. & x_{1,2} \vee x_{3,2} & [\ \mathit{CNF}\ (x_{1,2} \equiv \bot \Rightarrow x_{3,2} \equiv \top)\ ] \\
19. & x_{1,2} \vee \overline{x}_{1,3} & [\ \mathit{CNF}\ (x_{1,2} \equiv \bot \Rightarrow x_{1,3} \equiv \bot)\ ] \\
20. & x_{1,2} \vee \overline{x}_{2,3} & [\ \mathit{CNF}\ (x_{1,2} \equiv \bot \Rightarrow x_{2,3} \equiv \bot)\ ] \\
\begin{array}{l} 21. \\ 22. \end{array} & \left.\begin{array}{l} x_{1,2} \vee \overline{x}_{1,3} \\ x_{1,2} \vee \overline{x}_{2,3} \end{array}\right\} & [\ \mathit{CNF}\ (x_{1,2} \equiv \bot \Rightarrow C_3(3) \equiv \bot)\ ] \\
23. & (x_{1,2} \vee x_{2,3}) & [\ \text{By resolution of lines 8 and 21}\ ] \\
24. & x_{1,2} & [\ \mathit{CNF}\ (x_{1,2} \equiv \bot \Rightarrow \bot)\ ] \\
25. & 0 & [\ \mathit{CNF}\ (\bot)\ ]
\end{array}$$

Figure 6.8: Line-based dilemma refutation and resolution refutation of GT_3.

Proof: By proposition 5.6, it suffices to prove the theorem for equivalence-based dilemma. Let π_R be the resolution refutation of F constructed from the (equivalence-based) dilemma refutation π_D in the proof of theorem 6.20 (which can be done in polynomial time). Then it holds that

$$L(\pi_R) \leq 2 \cdot L(\pi_D) \cdot k^{2H(F)+3} \leq 2 \cdot L(\pi_D) \cdot k^{5H(F)} \tag{6.31}$$

and

$$W(\pi_R) \leq \big(2H(F)+3\big)\big(k-1\big) \leq 5kH(F) \tag{6.32}$$

(since $H(F) \geq 1$ for CNF formulas F without unit clauses). Also, by corollary 4.21 on page 58 we have

$$L(\pi_D) \geq 2^{H(F)/2} \tag{6.33}$$

(although this inequality is proved for binary formula relation dilemma, it is easy to verify that the same bound holds for equivalence-based NF-dilemma).

We want to show that π_R has size $S(\pi_R)$ polynomial in $S(\pi_D)$. Using the above inequalities, we get

$$\begin{aligned} S(\pi_R) &\leq L(\pi_R) \cdot W(\pi_R) \\ &\leq 2 \cdot L(\pi_D) \cdot k^{5H(F)} \cdot 5kH(F) && \text{[by (6.31) and (6.32)]} \\ &\leq 10 \cdot L(\pi_D)^2 \cdot 2^{H(F) \cdot 6 \log k} && \text{[since } 1 \leq H(F) \leq L(\pi_D)\text{]} \\ &\leq 10 \cdot L(\pi_D)^{2+12\log k} && \text{[by (6.33)]} \\ &\leq 10 \cdot S(\pi_D)^{2+12\log k} \end{aligned}$$

which is polynomial in $S(\pi_D)$ if k is fix. □

Note, however, that theorem 6.21 does *not* say that resolution p-simulates dilemma for k-CNF formulas. The theorem does not exclude the possibility that there are very deep but short dilemma proofs for which the translation to resolution in theorem 6.20 yields a superpolynomially long proof. The exact relation between resolution and (normal form) dilemma restricted to k-CNF formulas (i.e. are there p-simulations in either direction? are the proof systems p-equivalent? or incomparable?) is an open question, as is the relation of the two proof systems on general CNF formulas.

Another way of looking at theorem 6.20 is from the perspective of proof methods. As before, fix k and let F be an unsatisfiable k-CNF formula over n variables. Suppose that we parenthesize the formula in some way and feed it into Stålmarck's method, and that the parenthesized version of F has hardness degree $H(F) = h$. Then a trivial lower bound on the time needed to refute F with Stålmarck's method is $\Omega\left(n^h\right)$ (since the algorithm will have to perform at least one full iteration of (h-1)-saturation and this takes time at least n^h if F has n variables). According to theorem 6.20, for resolution we have $W_{\mathcal{R}}(F \vdash \perp) = \mathrm{O}(kh)$, so the minimum-width proof search algorithm discussed in section 3.1.3 finds a resolution refutation of F in time $n^{W_{\mathcal{R}}(F \vdash \perp)} \leq n^{ckh}$ for some constant c. Since k is fix, this implies that minimum-width proof search in resolution finds a refutation of a k-CNF formula F in time polynomial in the lower bound on the running time of Stålmarck's method.

The above reasoning can be generalized to propositional logic formulas as well. We have the following theorem.

Theorem 6.22 (Min-width proof search is polynomial in Stålmarck) *Suppose that F is a contradictory CNF formula in width $W(F) \leq k$ (for some fixed k) and that G is a tautology in propositional logic. Then*

1. *minimum-width proof search refutes F in time polynomial in the running time of Stålmarck's method.*

2. *minimum-width proof search proves G valid by refuting the Tseitin transformation G_t of G in time polynomial in the running time of Stålmarck's method on G.*

Proof: Part 1 of the theorem was proved above.

For part 2, suppose that G is a tautology of hardness $H_{\mathcal{D}}(G) = h$ in general propositional logic. Tseitin's transformation (figure 2.1 on page 19) applied on G returns a CNF formula G_t with $W(G_t) = \mathrm{O}(1)$. The original propositional logic formula G is exactly h-hard for binary dilemma, so G_t can be at most $h+ c'$-hard for NF-dilemma for some constant c' (show by induction that if $P \equiv Q$ is derived by propagation in a dilemma derivation π for G, then $x_P \equiv x_Q$ can be derived in at most constant extra depth in a corresponding derivation π_t for G_t). Since the number of variables in G_t is $|\mathit{Vars}(G_t)| = |\mathit{Sub}(G)|/2$, minimum-width proof search refutes G_t in time $|\mathit{Sub}(G)|^{\mathrm{O}(h)}$.

For Stålmarck's method, 0-saturation of $G^{\perp}$ takes time $\Omega\big(|\mathit{Sub}(G)|\big)$ (since all subformulas must be inspected at least once). Before (κ+1)-saturation terminates, the algorithm has to assume all subformulas in $\mathit{Sub}(G)$ true and false and for each pair of assumptions κ-saturate both branches. By induction, we get the lower bound on κ-saturation in Stålmarck's method $\Omega\big(|\mathit{Sub}(G)|^{\kappa+1}\big)$.

If the formula G is h-hard, Stålmarck's method must perform a full pass of (h-1)-saturation in time $\Omega\big(|\mathit{Sub}(G)|^{h}\big)$ before it can prove G valid and terminate. Thus minimum-width proof search is polynomial in Stålmarck's method as claimed. □

It is not known whether Stålmarck's proof method is polynomial in minimum-width proof search or whether there are formulas for which the latter algorithm is superpolynomially faster.

Regardless of what the relation between Stålmarck's method and minimum-width proof search might be, it should be stressed that theorem 6.22 is a *theoretical* result. Our main tools when comparing proof systems are polynomial simulations and superpolynomial separations, which both ignore polynomial differences. Needless to say, in practical applications polynomial factors *cannot* be neglected. In an applied context it might very well be the case that a polynomial algorithm, although theoretically appealing, is completely infeasible, while another algorithm with superpolynomial theoretical worst-case upper bound turns out to be the preferred choice because of empirically better running times. The question of what the practical implications of theorem 6.22 are (if there are any), therefore, must be settled by other means.

In section 3.3 we studied refutation length of random k-CNF formulas. Combining the upper bound in section 3.3.1 and the lower bound in section 3.3.2 with the translation from dilemma to resolution in theorem 6.20, we can derive bounds on the hardness degree in dilemma of uniformly random 3-CNF formulas (which can be generalized to k-CNF formulas for arbitrary $k \geq 3$ in the same way as the results in section 3.3).

We remind that the notation $F \sim \mathcal{F}_k^{n,\Delta}$ means that F is a uniformly random k-CNF formula with $m = \Delta n$ clauses on n variables and that θ_k^u is the threshold value such that for densities $\Delta \geq \theta_k^u$, formulas $F \sim \mathcal{F}_k^{n,\Delta}$ are unsatisfiable with probability $1 - \mathrm{o}(1)$ in n.

Theorem 6.23 (Bounds on hardness of random 3-CNF in dilemma) *Suppose that $F \sim \mathcal{F}_3^{n,\Delta}$, with $\Delta \geq \theta_3^u$. Then the following holds:*

1. *$H_{\mathcal{D}}(F) = \mathrm{O}(n/\Delta)$ with probability $1 - \mathrm{o}(1)$ in n,*
2. *$H_{\mathcal{D}}(F) = \Omega\left(n/\Delta^{2+\epsilon}\right)$ with probability $1 - \mathrm{o}(1)$ in n, where $\epsilon > 0$ is arbitrary.*

Proof: Let $F \sim \mathcal{F}_3^{n,\Delta}$ with $\Delta = m/n \geq \theta_3^u$.

1. By lemma 3.29, if a DLL procedure has branched over cn/Δ variables (for a suitably chosen constant c), then with probability $1 - \mathrm{o}(1)$ in n at least half of the remaining variables are critical (i.e. splitting over them leads to contradictions in both branches by unit propagation).

But then in particular there is at least *one* critical variable. It follows (by theorem 6.15) that it is sufficient to make $cn/\Delta + 1$ simultaneous assumptions to refute F in (binary) atomic reductio. Since atomic reductio is a subsystem of general dilemma, the hardness degree in general dilemma $H_{\mathcal{D}}(F)$ is at most $cn/\Delta + 1$ with the same probability $1 - \mathrm{o}(1)$ in n.

2. By theorem 3.36, the minimum width in which F can be refuted by resolution is

$$W_{\mathcal{R}}(F \vdash \bot) = \Omega\left(n/\Delta^{2+\epsilon}\right) \tag{6.34}$$

with probability $1 - \mathrm{o}(1)$ in n. Let the hardness degree of F in general NF-dilemma be h. Since $k = 3$ is fix, it follows from theorem 6.20 that

$$W_{\mathcal{R}}(F \vdash \bot) = \mathrm{O}(h). \tag{6.35}$$

Combining (6.34) and (6.35) and noting that the hardness degree in binary dilemma is bounded from below by the hardness degree in NF-dilemma, we get $H_{\mathcal{D}}(F) = \Omega\left(n/\Delta^{2+\epsilon}\right)$ with probability $1 - \mathrm{o}(1)$ in n, where ϵ is arbitrarily small but positive. □

We close this section with a supplementary remark to the exposition of the dilemma proof system in section 4.1.

Remark 6.24 (Dilemma hardness degree hierarchy) When we presented the dilemma proof system in chapter 4 and introduced the concept of hardness degree, we also raised the question whether there exists an infinite hierarchy of formulas with growing hardness or not. After a short discussion, we concluded without giving any formal proof that the infinite hardness degree hierarchy does indeed exist (section 4.1.6).

For readers not willing to accept the existence of this infinite hierarchy without a formal proof, we now show that such a proof follows easily from theorem 6.23. Fix $\Delta \geq \theta_3^u$ and let n grow. Then by theorem 6.23 we get that for 3-CNF formulas $F \sim \mathcal{F}_3^{n,\Delta}$, $H_{\mathcal{D}}(F_n) = \Omega(n)$ with probability $1 - \mathrm{o}(1)$ in n. In particular, since $H_{\mathcal{D}}(F_n) = \Omega(n)$ with high probability, there must exist examples of formulas F_n with growing hardness degree. This proves the existence of the infinite hardness degree hierarchy in the dilemma proof system.

Chapter 7

Conclusions

In this chapter we summarize the results of our work. This Master's thesis is quite long (one might even say a little on the long side), and a considerable part of it contains accounts of earlier research. Here, we highlight our contributions to the subject and try to put our results in perspective. We also discuss some of the questions that remain open and give suggestions for further research.

7.1 Summary of Results

The results presented in this section are divided into those concerning the dilemma and RAA proof systems (section 7.1.1) and those relating these proof systems to resolution (section 7.1.2).

7.1.1 Results for Dilemma and Reductio

The main part of this thesis is dedicated to the formalization of the dilemma proof system with subsystems (chapter 4 and appendix A) and the development of tools for proving bounds on different proof systems in the "dilemma proof system family" (chapter 5). While this is perhaps not an original contribution in the strict sense, to our knowledge it is the first complete explicit formal exposition of the dilemma proof system. This formalization of dilemma was a necessary prerequisite for being able to prove the results listed below.

Our conclusions from the comparative studies of different dilemma and RAA proof systems are:

- Dilemma (whether defined as general dilemma $\mathcal{D}$ or the bivalent subsystem $\mathcal{D}_B$ used in Stålmarck's method) is a quasi-automatizable propositional proof system (theorem 4.35 on page 78).
- Bivalent dilemma $\mathcal{D}_B$ is linearly stronger than atomic dilemma $\mathcal{D}_A$ with respect to proof hardness for general propositional logic formulas (theorem 6.1 on page 93) and almost certainly for CNF formulas as well (conjecture 6.2 on page 95). Our separation with respect to hardness does not yield any separation with respect to proof length and size, however, which shows that hardness degree in atomic dilemma is not a meaningful measure (theorem 6.4 on page 98).

- Bivalent dilemma $\mathcal{D}_B$ simulates general dilemma $\mathcal{D}$ restricted to minimum-depth proofs polynomially with respect to proof length and size and is at most quasi-polynomially worse than $\mathcal{D}$ with respect to size for proofs in arbitrary depth (theorem 6.3 on page 97).

- Atomic dilemma $\mathcal{D}_A$ is linearly stronger than atomic reductio $\mathcal{RAA}_A$ with respect to proof hardness and exponentially stronger with respect to proof length and size (theorem 6.7 on page 102 and corollary 6.19 on page 107).

- Bivalent reductio $\mathcal{RAA}_B$ is linearly stronger than atomic reductio $\mathcal{RAA}_A$ with respect to proof hardness and exponentially stronger with respect to proof length and size (corollary 6.8 on page 102).

- Bivalent reductio $\mathcal{RAA}_B$ can be separated from atomic dilemma $\mathcal{D}_A$ linearly with respect to hardness degree (corollary 6.9 on page 102).

- General reductio $\mathcal{RAA}$ is at most quasi-polynomially worse than general dilemma $\mathcal{D}$ with respect to proof size (theorem 6.12 on page 103).

7.1.2 Results for Dilemma and Resolution

The results about the comparative proof power of different dilemma and RAA proof systems are mainly of theoretical value, in that they help us understand better the "inner workings" of the dilemma proof system and Stålmarck's method. More interesting from an applied perspective are the comparisons of dilemma and resolution. Results about the proof system underlying a proof search algorithm can give us theoretically proven bounds on the performance of the algorithm. Since the dilemma and resolution proof systems both lie at the foundations of widely used proof methods, such bounds can potentially be of great interest not only theoretically but also in practice.

Most of the results relating dilemma and resolution have been proven for the normal form dilemma proof system (NF-dilemma for short) defined in section 5.2.1 for CNF (and other types of) formulas. As was discussed in that section, lower bounds on refuting CNF formulas in NF-dilemma imply at least the same bounds in ordinary binary dilemma.

Below follows a list of our results for dilemma and resolution.

- Derivation *depth* in dilemma corresponds to derivation *width* in resolution. More precisely, a general NF-dilemma refutation in depth d and length L of a CNF formula F with maximum clause width k can be translated to a resolution refutation of F in width $\mathrm{O}(kd)$ and length $\left(Lk^d\right)^{\mathrm{O}(1)}$ (theorem 6.20 on page 107).

- For k-CNF formulas with k fix, resolution p-simulates general NF-dilemma restricted to minimum-depth proofs (theorem 6.21 on page 109).

- For propositional logic tautologies the minimum-width proof search algorithm suggested by Ben-Sasson and Wigderson [7] combined with Tseitin's transformation to CNF (figure 2.1) has running time polynomial in the running time of Stålmarck's method (theorem 6.22 on page 112).

- If F is a uniformly random 3-CNF formula with Δn clauses on n variables (where the density Δ is sufficiently large so that F is almost certainly unsatisfiable), then with probability $1 - \mathrm{o}(1)$ in n it holds for the hardness degree $H_{\mathcal{D}}(F)$ of F in (binary) general dilemma that
$$\Omega\left(n/\Delta^{2+\epsilon}\right) \leq H_{\mathcal{D}}(F) \leq \mathrm{O}(n/\Delta),$$
where ϵ is arbitrarily small but positive (theorem 6.23 on page 113).

- Atomic NF-reductio and tree-like resolution are p-equivalent with respect to proof length (theorem 6.15 on page 105).

- Atomic NF-dilemma and bivalent NF-reductio are exponentially stronger than tree-like resolution with respect to proof length (corollary 6.18 on page 107). Since the formulas used in the separation have constant hardness in *binary* dilemma, this shows that there are formulas where the (atomic) Stålmarck method will find a proof in polynomial time but all DLL procedures, no matter what the splitting rule, take exponential time.

7.2 Open Questions

In this section, we list the main open questions mentioned in chapter 6 and give suggestions for further research.

- We have shown that for k-CNF formulas with k fix, resolution p-simulates dilemma restricted to minimum-depth proofs, but what is the relation between (normal form) dilemma without this minimum-depth restriction and resolution on k-CNF formulas? And what is the relation between dilemma and resolution on general CNF formulas?

- We have shown that bounds on depth in dilemma translates into bounds on width in resolution. Is this true in the opposite direction as well? That is, can resolution in width w be transformed to dilemma in depth $\mathrm{O}(w)$? If the converse of theorem 6.20 could be proven true, this would show that at least for k-CNF formulas, Stålmarck's method and minimum-width proof search are basically the same algorithm but in different proof systems. It might just as well be the case, however, that there are formula families which can be refuted in fix width in resolution but have non-constant refutation depth in dilemma. One candidate for closer study could be the separations of general resolution from DP-resolution in [10]. It has not been possible to do this whithin the framework of this Master's project.

- In sections 6.1 and 6.2 we explored the relations between different dilemma and reductio proof systems. As has already been noted, this study is mainly of theoretical interest and is unlikely to have any practical consequences. Nevertheless, we list below the questions remaining if we want to get a complete picture of the relations between the proof subsystems.
  - What is the relationship between bivalent and general dilemma with respect to proof length and size?
  - What is the relationship between general and bivalent reductio with respect to proof hardness, length and size?

  - Can bivalent dilemma and bivalent reductio be separated superpolynomially with respect to proof length and size or superlogarithmically with respect to proof hardness? (That is, can the result in [53] be strengthened?)
  - What is the relationship between general dilemma and general reductio with respect to proof length and size?
  - Can bivalent reductio and atomic dilemma be separated with respect to proof length and size? Are the two proof systems incomparable?

Finally, we mention two questions of a more practical nature.

- We have shown that minimum-width proof search in resolution is polynomial in Stålmarck's method, but how do the two algorithms compare in practice? It would be interesting to make an efficient implementation of minimum-width proof search and benchmark it against Stålmarck's method for (a) k-CNF or general CNF formulas which arise in practical problems and (b) fixed-width CNF formulas obtained by transforming general propositional logic formulas arising in practical problems. Such benchmarks could reveal whether minimum-width proof search is a viable alternative to Stålmarck's method when it comes to real-life problems and also shed some light on the relation between the hardness measures of proof depth in dilemma and proof width in resolution.

- What is the exact relation between bivalent and general dilemma with respect to hardness? Of course the hardness degrees are within a constant and thus from a theoretical point of view $\mathcal{D}_B$ and $\mathcal{D}$ are equivalent with respect to this measure. Still, from an applied perspective it would be interesting if one could separate general and bivalent dilemma with respect to hardness degree in the sense of finding a family of polynomial-size formulas F_n such that $H_{\mathcal{D}}(F_n) = n$ but $H_{\mathcal{D}_B}(F_n) \geq c \cdot n$ for some $c > 1$. This could show that there are formulas for which the general Stålmarck method is better than the bivalent Stålmarck method. But then for all we know it might be the case that $H_{\mathcal{D}}(F) \leq H_{\mathcal{D}_B}(F) \leq H_{\mathcal{D}}(F) + c$ for all formulas F. If so, the bivalent Stålmarck method is likely to be better than the general Stålmarck method not only empirically but also theoretically.

Bibliography

[1] Miklós Ajtai. The complexity of the pigeonhole principle. In *Proceedings 29th Annual IEEE Symposium on Foundations of Computer Science (FOCS '88)*, pages 346–355, White Plains, New York, 24–26 October 1988.

[2] Paul Beame, Richard Karp, Michael Saks, and Toniann Pitassi. On the complexity of unsatisfiability proofs of random k-CNF formulas. In *Proceedings 30th Annual ACM Symposium on Theory of Computing (STOC '98)*, pages 561–571, New York, 23–26 May 1998.

[3] Paul Beame, Richard Karp, Michael Saks, and Toniann Pitassi. *The Efficiency of Resolution and Davis-Putnam Procedures.* Submitted for publication, May 1999.

[4] Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *Proceedings 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS '96)*, pages 274–282, Burlington, Vermont, 14–16 October 1996.

[5] 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.

[6] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. *Near-Optimal Separation of Treelike and General Resolution.* Report TR00-005, Electronic Colloquium on Computational Complexity, January 2000. Available at `http://www.eccc.uni-trier.de/eccc/`.

[7] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. In *Proceedings 31st Annual ACM Symposium on Theory of Computing (STOC '99)*, pages 517–526, Atlanta, Georgia, 1–4 May 1999.

[8] Per Bjesse. *Gate Level Description of Synchronous Hardware and Automatic Verification Based on Theorem Proving.* PhD thesis, Chalmers University of Technology and Göteborg University, May 2001. Available at `http://www.cs.chalmers.se/~bjesse/`.

[9] Robin E. Bloomfield and Dan Craigen. *Formal Methods Diffusion: Past Lessons and Future Prospects.* Report, Adelard, London, December 1999. Available at `http://www.adelard.co.uk/resources/fmreport/`.

[10] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. Exponential separations between restricted resolution and cutting planes

proof systems. In *Proceedings 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS '98)*, pages 638–647, Los Alamitos, California, 8–11 November 1998.

[11] Maria Luisa Bonet and Nicola Galesi. A study of proof search algorithms for resolution and polynomial calculus. In *Proceedings 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS '99)*, New York City, New York, 17–19 October 1999.

[12] Randall E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[13] Randall E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24:293–318, 1992.

[14] Samuel R. Buss, editor. *Handbook of Proof Theory*. Elsevier Science, Amsterdam, 1998.

[15] Vašek Chvátal and Bruce A. Reed. Mick gets some (the odds are on his side). In *Proceedings 33rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '92)*, pages 620–627, Pittsburgh, Pennsylvania, 24–27 October 1992.

[16] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[17] Edmund M. Clarke and Jeanette M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, December 1996.

[18] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pages 174–183, Philadelphia, Pennsylvania, 22–24 May 1996.

[19] Stephen Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36ff, 1979.

[20] Thomas H. Cormen, Charles E. Leiserson, and Ronald R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1991.

[21] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings 11th National Conference on Artificial Intelligence*, pages 21–27, Menlo Park, California, 1993. AAAI Press.

[22] Dirk van Dalen. *Logic and Structure*. Universitext. Springer-Verlag, Berlin, 3rd, augmented edition, 1994.

[23] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[24] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[25] Ehud Friedgut. Sharp thresholds of graph properties, and the k-SAT problem (with appendix by Jean Bourgain). *Journal of the American Mathematical Society*, 12:1017–1054, 1999.

[26] Alan Frieze and Stephen Suen. Analysis of two simple heuristics on a random instance of k-SAT. *Journal of Algorithms*, 20(2):312–355, March 1996.

[27] Xudong Fu. *On The Complexity of Proof Systems.* PhD thesis, University of Toronto, 1995.

[28] Gerhard Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen.* North-Holland, 1969. Translated from 1935 original in German.

[29] James Gleick. A bug and a crash. *New York Times Magazine*, 1 December 1996. Available at `http://www.around.com/`.

[30] Andreas Goerdt. A threshold for unsatisfiability. *Journal of Computer and System Sciences*, 53(3):469–486, December 1996.

[31] Aarti Gupta. Formal hardware verification methods: A survey. *Journal of Formal Methods in System Design*, 1:151–238, 1992.

[32] John Harrison. Stålmarck's algorithm as a HOL derived rule. In *Proceedings 9th International Conference on Theorem Proving in Higher Order Logics (TPHOL '96)*, volume 1125 of *Lecture Notes in Computer Science*, pages 221–234. Springer-Verlag, 1996.

[33] Arndt Jonasson. *A Result on the U_n Formulas.* Draft, Logikkonsult NP U-96037, December 1997.

[34] Alexis C. Kaporis, Lefteris M. Kirousis, Yannis C. Stamatiou, Malvina Vamvakari, and Michele Zito. The unsatisfiability threshold revisited. In *Electronic Notes in Discrete Mathematics*, volume 9. Elsevier Science Publishers, 2001.

[35] Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random Boolean expressions. *Science*, 264(5163):1297–1301, 27 May 1994.

[36] Lefteris M. Kirousis, Evangelos Kranakis, and Danny Krizanc. Approximating the unsatisfiability threshold of random formulas (extended abstract). In *Proceedings 4th Annual European Symposium on Algorithms (ESA '96)*, volume 1136 of *Lecture Notes in Computer Science*, pages 27–38. Springer-Verlag, 1996.

[37] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Yannis C. Stamatiou. Approximating the unsatisfiability threshold of random formulas. *Random Structures and Algorithms*, 12(3):253–269, May 1998.

[38] Tracy Larrabee and Yumi Tsuji. *Evidence for a Satisfiability Threshold for Random 3-CNF Formulas.* Technical Report UCSC-CRL-92-42, University of California, Santa Cruz, November 1992. Available at `http://citeseer.nj.nec.com/`.

[39] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *Proceedings 10th National Conference on Artificial Intelligence*, pages 459–465. MIT Press, July 1992.

[40] Joachim Parrow. *Trollkarlen blir Ingenjör – Formella Metoder: En Översikt*, May 1997. In Swedish. Available at `http://www.sics.se/~joachim/`.

[41] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[42] *The Pentium Papers: A collection of documents associated with the Pentium floating point division bug*, December 1994. Available at `http://www-europe.mathworks.com/company/pentium/`.

[43] Dag Prawitz. *Natural Deduction, A Proof-Theoretical Study.* Almqvist & Wiksell, 1965.

[44] Prover Technology AB. *Prover 4.0 Application Programming Interface Reference Manual*, 2000. PPI-01-ARM-1.

[45] Prakash Rashinkar, Peter Paterson, and Leena Singh. *System-on-a-chip verification: methodology and techniques.* Kluwer Academic Publishers, Boston, 2001.

[46] *Requirements for Safety Related Software in Defence Equipment.* Standard DEF STAN 00-55, Ministry of Defence, United Kingdom, August 1997. Available at `http://www.dstan.mod.uk/`.

[47] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[48] Carl-Johan Seger. *An Introduction to Formal Hardware Verification.* Technical Report TR-92-13, University of British Columbia, June 1992. Available at `ftp://ftp.cs.ubc.ca/pub/local/techreports/`.

[49] Mary Sheeran and Gunnar Stålmarck. A tutorial on Stålmarck's proof procedure for propositional logic. *Formal Methods in System Design*, 16(1):23–58, January 2000.

[50] Raymond M. Smullyan. *First-Order Logic.* Springer-Verlag, Berlin, 2nd edition, 1971.

[51] Gunnar Stålmarck. *Personal communication.*

[52] Gunnar Stålmarck. A note on the computational complexity of the pure classical implication calculus. *Information Processing Letters*, 31(6):277–278, June 1989.

[53] Gunnar Stålmarck. *The REDUCTIO Hardness of the U_n-formulas.* Draft, May 1996.

[54] Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33(3):277–280, May 1996.

[55] Gunnar Stålmarck and Mårten Säflund. Modelling and verifying systems and software in propositional logic. In *Proceedings Safety of Computer Control Systems (SAFECOMP '90)*, pages 31–36. Pergamon Press, 1990.

[56] Grigori Tseitin. On the complexity of derivation in propositional calculus. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning*, pages 466–483. Springer-Verlag, Berlin, 1983.

[57] Alasdair Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1(4):425–467, 1995.

[58] Filip Widebäck. *Stålmarck's Notion of n-Saturation.* Technical Report NP-K-FW-200, Logikkonsult NP AB, Stockholm, January 1996.

All electronic documents last visited September 7, 2001.

Appendix A

Equivalence-Based Dilemma

In this appendix we give a full description of the modification of the dilemma proof system introduced in section 5.2.2 and used in some of the proofs in chapter 6. For lack of better names, we call the variant of dilemma presented here “equivalence-based dilemma” and the original version in chapter 4 “formula relation dilemma”.

In equivalence-based dilemma we do not have formula relations containing equivalence classes. Instead, we add rules for formal manipulation of formula equivalences. The reason for this is that it is a way to achieve a closer correspondence between dilemma and resolution. Using a variant of dilemma with explicit derivations of all equivalences between formulas makes the proofs of the theorems in chapter 6 comparing the two proof systems considerably easier.

This appendix is organized as follows: We start by giving rules for formula equivalences in section A.1. In section A.2 we list the propagation rules (which are identical for equivalence-based and formula relation dilemma). The dilemma rule is presented and its different variants discussed in section A.3, after which a formal definition of what constitutes an equivalence-based dilemma derivation is given in section A.4. Finally, the differences between equivalence-based and formula relation dilemma and their implications for the results presented in this Master’s thesis are analyzed in section A.5. Throughout the appendix, the letters P, Q, R and S (with or without indices) denote arbitrary logical formulas and R (with or without index) denotes an arbitrary formula relation.

A.1 Formula Equivalence Rules

The dilemma proof system rules for formal manipulation of formula equivalences are given in figure A.1 on the next page. These rules encode the fact that $\equiv$ is an equivalence relation, i.e. reflexive (rule (E1)), commutative (rule (E2)), and transitive (rules (E3), (E4) and (E5)), which in addition respects complement (rules (E6), (E7) and (E8)). The introduction rules (E9) and (E10) can be considered as special cases of rule (E3). In the formula relation dilemma proof system none of these rules are needed, since the corresponding information is already available in the equivalence classes of a formula relation.

In addition to the rules mentioned above, there is also a rule (E11) for deriving a contradiction (corresponding to rule (4.18) on page 48). The interpretation

$$\frac{}{P \equiv P} \tag{E1}$$

$$\frac{P \equiv Q}{Q \equiv P} \tag{E2}$$

$$\frac{P \equiv Q \quad Q \equiv R}{P \equiv R} \tag{E3}$$

$$\frac{P \equiv Q \quad Q \equiv \top}{P \equiv \top} \tag{E4}$$

$$\frac{P \equiv Q \quad Q \equiv \bot}{P \equiv \bot} \tag{E5}$$

$$\frac{P \equiv Q}{P^C \equiv Q^C} \tag{E6}$$

$$\frac{P \equiv \top}{P^C \equiv \bot} \tag{E7}$$

$$\frac{P \equiv \bot}{P^C \equiv \top} \tag{E8}$$

$$\frac{P \equiv \top \quad Q \equiv \top}{P \equiv Q} \tag{E9}$$

$$\frac{P \equiv \bot \quad Q \equiv \bot}{P \equiv Q} \tag{E10}$$

$$\frac{P \equiv P^C}{\bot} \tag{E11}$$

Figure A.1: Dilemma proof system rules for formula equivalences.

of this last rule is that if we have derived from a set of assumptions that a formula has the same truth value as its complement, we can conclude that the set of assumptions is contradictory. We overload the symbol $\bot$ to denote (when appearing alone on a line) that a contradiction has been derived. It thus corresponds to the canonical contradictory formula relation $\bot_{\mathsf{R}}$ consisting of only one equivalence class.

A.2 Propagation Rules

In this section we list the dilemma proof system propagation rules for the binary logical connectives conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) and bi-implication ($\leftrightarrow$). As was mentioned above, the propagation rules are the same for equivalence-based and formula relation dilemma. Note that due to the definition of complement, no special rules for negation ($\neg$) are needed.

$$\frac{P \wedge Q \equiv \top}{P \equiv \top} \qquad \frac{P \wedge Q \equiv \top}{Q \equiv \top} \tag{C1}$$

$$\frac{P \wedge Q \equiv P^C}{P \equiv \top} \qquad \frac{P \wedge Q \equiv Q^C}{Q \equiv \top} \tag{C2}$$

$$\frac{P \wedge Q \equiv P^C}{Q \equiv \bot} \qquad \frac{P \wedge Q \equiv Q^C}{P \equiv \bot} \tag{C3}$$

$$\frac{P \equiv \top}{P \wedge Q \equiv Q} \qquad \frac{Q \equiv \top}{P \wedge Q \equiv P} \tag{C4}$$

$$\frac{P \equiv \bot}{P \wedge Q \equiv \bot} \qquad \frac{Q \equiv \bot}{P \wedge Q \equiv \bot} \tag{C5}$$

$$\frac{P \equiv Q}{P \wedge Q \equiv P} \qquad \frac{P \equiv Q}{P \wedge Q \equiv Q} \tag{C6}$$

$$\frac{P \equiv Q^C}{P \wedge Q \equiv \bot} \tag{C7}$$

Figure A.2: Dilemma proof system rules for conjunction.

$$\frac{P \vee Q \equiv \bot}{P \equiv \bot} \qquad \frac{P \vee Q \equiv \bot}{Q \equiv \bot} \tag{D1}$$

$$\frac{P \vee Q \equiv P^C}{P \equiv \bot} \qquad \frac{P \vee Q \equiv Q^C}{Q \equiv \bot} \tag{D2}$$

$$\frac{P \vee Q \equiv P^C}{Q \equiv \top} \qquad \frac{P \vee Q \equiv Q^C}{P \equiv \top} \tag{D3}$$

$$\frac{P \equiv \top}{P \vee Q \equiv \top} \qquad \frac{Q \equiv \top}{P \vee Q \equiv \top} \tag{D4}$$

$$\frac{P \equiv \bot}{P \vee Q \equiv Q} \qquad \frac{Q \equiv \bot}{P \vee Q \equiv P} \tag{D5}$$

$$\frac{P \equiv Q}{P \vee Q \equiv P} \qquad \frac{P \equiv Q}{P \vee Q \equiv Q} \tag{D6}$$

$$\frac{P \equiv Q^C}{P \vee Q \equiv \top} \tag{D7}$$

Figure A.3: Dilemma proof system rules for disjunction.

$$\frac{P \to Q \equiv \bot}{P \equiv \top} \tag{I1}$$

$$\frac{P \to Q \equiv \bot}{Q \equiv \bot} \tag{I2}$$

$$\frac{P \to Q \equiv P}{P \equiv \top} \tag{I3}$$

$$\frac{P \to Q \equiv P}{Q \equiv \top} \tag{I4}$$

$$\frac{P \to Q \equiv Q^C}{P \equiv \bot} \tag{I5}$$

$$\frac{P \to Q \equiv Q^C}{Q \equiv \bot} \tag{I6}$$

$$\frac{P \equiv \top}{P \to Q \equiv Q} \tag{I7}$$

$$\frac{Q \equiv \top}{P \to Q \equiv \top} \tag{I8}$$

$$\frac{P \equiv \bot}{P \to Q \equiv \top} \tag{I9}$$

$$\frac{Q \equiv \bot}{P \to Q \equiv P^C} \tag{I10}$$

$$\frac{P \equiv Q}{P \to Q \equiv \top} \tag{I11}$$

$$\frac{P \equiv Q^C}{P \to Q \equiv P^C} \qquad \frac{P \equiv Q^C}{P \to Q \equiv Q} \tag{I12}$$

Figure A.4: Dilemma proof system rules for implication.

$$\frac{P \leftrightarrow Q \equiv \top}{P \equiv Q} \tag{B1}$$

$$\frac{P \leftrightarrow Q \equiv \bot}{P \equiv Q^C} \tag{B2}$$

$$\frac{P \leftrightarrow Q \equiv P}{Q \equiv \top} \qquad \frac{P \leftrightarrow Q \equiv Q}{P \equiv \top} \tag{B3}$$

$$\frac{P \leftrightarrow Q \equiv P^C}{Q \equiv \bot} \qquad \frac{P \leftrightarrow Q \equiv Q^C}{P \equiv \bot} \tag{B4}$$

$$\frac{P \equiv \top}{P \leftrightarrow Q \equiv Q} \qquad \frac{Q \equiv \top}{P \leftrightarrow Q \equiv P} \tag{B5}$$

$$\frac{P \equiv \bot}{P \leftrightarrow Q \equiv Q^C} \qquad \frac{Q \equiv \bot}{P \leftrightarrow Q \equiv P^C} \tag{B6}$$

$$\frac{P \equiv Q}{P \leftrightarrow Q \equiv \top} \tag{B7}$$

$$\frac{P \equiv Q^C}{P \leftrightarrow Q \equiv \bot} \tag{B8}$$

Figure A.5: Dilemma proof system rules for bi-implication.

Because of the closure under transitivity and complement, the rules given in this section form a complete set of propagation rules for formula relation dilemma. For equivalence-based dilemma the matter is slightly less straightforward. When we replace formula relations with the derivation rules for formula equivalence in section A.1, we also enlarge the set of distinct non-trivial propagation rules. The new rules are redundant in the sense that they can be discarded without affecting the derivation depth, since the same conclusions can be reached by using our chosen set of propagation rules and the rules for formula equivalence in figure A.1. They do affect proof length and size in equivalence-based dilemma, however, but only by a constant factor.

In order to minimize the number of cases that need to be analyzed in proofs about dilemma in chapter 6, we have chosen not to include the full set of possible propagation rules for equivalence-based dilemma. The verification of the statement that the omitted rules only affect proof length and size in equivalence-based dilemma by a constant factor is a routine matter and is left to the reader in order to save space in this report.

Symmetric instances of propagation rules are considered the same and are therefore listed on the same line in the figures presenting the propagation rules. Also, each rule always has exactly one consequence, which is why certain rules have identical premises (as for instance rules (C2) and (C3)).

The dilemma proof system rules for conjunction are given in figure A.2 and the rules for disjunction in figure A.3. The propagation rules for implication are presented in figure A.4. Since implication is not symmetric, these rules look rather different than those for the other connectives. To avoid confusion about formula relations and formula equivalences on the one hand (for which we use the notation $\equiv$) and the *logical connective* equivalence or bi-implication (denoted $\leftrightarrow$), we will refer to the connective $\leftrightarrow$ exclusively as "bi-implication". Needless to say, equivalence relations and bi-implication are very intimately connected. This is illustrated by the dilemma proof system rules given in figure A.5. Note that all rules for bi-implication are invertible.

A.3 The Dilemma Rule

In addition to the rules for equivalence in section A.1 and the propagation rules in section A.2, a rule for branching is needed in order to make the proof system complete for propositional logic. The dilemma proof system is named after the special "branch-and-merge" rule, called the *dilemma rule*, used in the proof system. It is this rule (in combination with the large set of propagation rules) that gives the dilemma proof system its characteristic properties.

If P, Q and R are arbitrary logical formulas (where we temporarily abuse our notation by allowing $Q = \top/\bot$), $\psi_i \in \{P \equiv R, P \equiv R^C, P \equiv \top, P \equiv \bot, \bot\}$ denote arbitrary equivalences (or contradiction) and π_i, $i = 1, 2, 3$ stand for dilemma derivations (which will be formally defined in section A.4), the dilemma rule in its most general form can be pictured as:

$$\cfrac{\begin{array}{cc} \multicolumn{2}{c}{\pi_1} \\ \hline P \equiv Q & P \equiv Q^C \\ \pi_2 & \pi_3 \\ \psi_1 & \psi_1 \\ \vdots & \vdots \\ \psi_k & \psi_k \end{array}}{\begin{array}{c} \psi_1 \\ \vdots \\ \psi_k \end{array}} \tag{A.1}$$

The interpretation of this rule is that if we can derive the equivalences $\psi_1, \ldots, \psi_k$ both under the assumption $P \equiv Q$ and under the complementary assumption $P \equiv Q^C$, then we can conclude that $\psi_1, \ldots, \psi_k$ hold regardless of the truth values of P and Q.

As a special case, if a contradiction is reached in one of the branches, we can conclude that all equivalences derived in the other branch hold. After simplification, we get the two symmetric special forms of the dilemma rule

$$\cfrac{\begin{array}{cc} \multicolumn{2}{c}{\pi_1} \\ \hline P \equiv Q & P \equiv Q^C \\ \pi_2 & \\ \bot & \end{array}}{P \equiv Q^C} \tag{A.2}$$

for a contradiction in the left branch and

$$\begin{array}{c} \pi_1 \\ \hline P \equiv Q \qquad \begin{array}{c} P \equiv Q^C \\ \pi_3 \\ \bot \end{array} \\ \hline P \equiv Q \end{array} \tag{A.3}$$

for a contradiction in the right branch.

When reasoning about the dilemma rule, we call the equivalences $P \equiv Q$ and $P \equiv Q^C$ in the rule schemas above *dilemma rule assumptions* and instances of rules (A.1), (A.2) and (A.3) *dilemma rule applications*.

A.4 Formal Description of the Proof System

We are now ready to define more formally what we mean by a derivation in the equivalence-based dilemma proof system given by the equivalence relation rules in figure A.1, the propagation rules in figures A.2, A.3, A.4 and A.5 and the dilemma rules (A.1), (A.2) and (A.3).

Intuitively, when defining the equivalence-based version of dilemma we want to mimic the definitions given in chapter 4. Unfortunately, it turns out that while gaining in convenience by abandoning the use of formula relations, we lose the elegance of definition 4.9 on page 50. The problem is that in formula relation dilemma, the current formula relation contains all formula equivalences derived up to that point in the derivation. In equivalence-based dilemma, these equivalences can be scattered throughout the preceding part of the derivation, and we need to determine which of the earlier equivalences can be used at a certain point to derive new equivalences and which cannot.

To get around this problem, we first describe what a dilemma derivation *looks like*, somewhat awkwardly dodging the question how to determine whether a derivation is legal or not. Once the structure of a dilemma derivation is clear, we use information about the derivation structure to define the concepts of *scope* and *depth*. Having defined these concepts, we finally turn to the question what constitutes a valid equivalence-based dilemma derivation.

Definition A.1 (Dilemma derivation structure) *From a structural point of view, an equivalence-based dilemma derivation is a sequence of (possibly interleaved) formula equivalences ($P \equiv Q$, $P \equiv Q^C$, $P \equiv \top$, $P \equiv \bot$ and $\bot$) and dilemma rule applications* (A.1), (A.2) *and* (A.3). *Recursively, the branches of a dilemma rule application are themselves dilemma derivations.*

A dilemma derivation is always defined relative to some formula F in the sense that the only admissible logical formulas in the derivation are subformulas of F and their negations (i.e. the elements in $\mathit{Sub}\,(F)$). In other words, the dilemma proof system obeys the *subformula principle* (definition 2.21 on page 16).

Although we have not yet addressed the question how to determine in what ways formula equivalences and dilemma rule applications may be legally inserted into a derivation, the structure of a dilemma derivation is now clear.

Next, we define the *scope* of dilemma rule assumptions and dilemma rule applications.

Definition A.2 (Scope) *The* scope *of a dilemma rule assumption is all formula equivalences occurring in the branch of the assumption (including the assumption itself and all formula equivalences in nested dilemma rule applications) up to the point where the two branches of the dilemma rule application are merged.*

The scope of a dilemma rule application is the combination of the scopes of the two complementary assumptions of the dilemma rule application.

Thus, for example, the scope of the dilemma rule assumption $P \equiv Q$ in (A.1) is $P \equiv Q$ itself, the formula equivalences in π_2 and the equivalences $\psi_1, \ldots, \psi_k$.

Definition A.3 (Derivation depth) *A formula equivalence in a derivation that is not within the scope of any application of the dilemma rule has* derivation depth *(or simply* depth*) zero. A formula equivalence which is within the scope of j applications of the dilemma rule has derivation depth j.*

The derivation depth of a dilemma derivation is the maximum depth of an equivalence in the derivation.

Using the concepts of scope and depth, we can determine which formula equivalences are admissible at a point in a dilemma derivation as premises for deriving further equivalences. We call such admissible formula equivalences *active equivalences*.

Definition A.4 (Active equivalence) *A dilemma rule assumption $P \equiv Q$ is an* active assumption *at a point in a dilemma derivation if the formula equivalence at that point is within the scope of the assumption $P \equiv Q$.*

At a given point in a dilemma derivation, a formula equivalence occurring earlier in the derivation is an active equivalence *if it is either a zero-depth formula equivalence or is within the scope of an active dilemma rule assumption but not within the scope of a no longer active (i.e. closed) nested dilemma rule application.*

Note that by definition A.2, all active assumptions are also active equivalences.

We are now finally ready to formally state the rules according to which a legal equivalence-based dilemma derivation is formed.

Definition A.5 (Equivalence-based dilemma derivation) *Let F be an arbitrary formula in propositional logic.*

An equivalence-based dilemma derivation *from F is a sequence of formula equivalences and dilemma rule applications that obeys the subformula principle and is formed according to the rules below.*

A dilemma proof *of F starts with the equivalence $F \equiv \bot$ and shows that F is a tautology by deriving $\bot$ (which corresponds to starting with the formula relation $F^{\bot}$ and deriving a contradictory formula relation).*

A dilemma refutation *of F starts with the formula equivalence $F \equiv \top$ (which corresponds to the formula relation $F^{\top}$) and proves that F is contradictory by deriving $\bot$ from this equivalence.*

Let $P, Q \in Sub\,(F)$. At any given point in a dilemma derivation, new formula equivalences can be added to the derivation according to the following rules:

Repetition *At any point in a dilemma derivation, any active formula equivalence* $P \equiv Q$ *(as defined in definition A.4) may be restated (this rule is included for purely practical reasons to simplify the definition of the dilemma rule.)*

Propagation *If at a point in a dilemma derivation there is among the rules in figures A.1, A.2, A.3, A.4 and A.5 a rule*

$$\frac{P_1 \equiv Q_1, \ldots, P_j \equiv Q_j}{P \equiv Q}$$

for which all premises $P_1 \equiv Q_1, \ldots, P_j \equiv Q_j$ *are among the active equivalences, then the formula equivalence* $P \equiv Q$ *can be added at that point in the derivation. If* $P \equiv P^C$ *is an active equivalence the symbol* $\perp$ *may be added in accordance with rule* (E11).

Branching *At any point in a dilemma derivation, a dilemma rule application may be introduced by branching over the truth or falsity of a formula equivalence* $P \equiv Q$*. The two branches of a dilemma rule application recursively obey the same derivation rules as the dilemma derivation itself.*

Merging *At any point in a dilemma derivation, the innermost open dilemma rule application may be closed according to the dilemma rule schemes* (A.1), (A.2) *or* (A.3). *(Note that in order to simplify the formal description of the dilemma rule applications, we add the restriction that the branches of a dilemma rule application should end with a list of the formula equivalences that we want to deduce, possibly including repetitions of equivalences derived earlier in the branches.)*

A dilemma proof or refutation ends when $\perp$ *has been derived at zero depth according to the rules above.*

A.5 Equivalence Rules vs. Formula Relations

The relations between dilemma and resolution in chapter 6 are shown for the equivalence-based variant of the proof system formally defined in definition A.5 above, while Stålmarck's proof search algorithm is based on formula relation dilemma. Because of this, it seems appropriate to comment on how our results on depth, length and size of dilemma proofs are affected by the differences between equivalence-based and formula relation dilemma.

It follows from our discussion in section A.2 that the hardness degree of a formula is not affected by the choice between equivalence-based and formula relation dilemma. As to derivation length and size, it is not too difficult to see that the two dilemma proof system variants are polynomially related. We give informal proofs of these facts below.

It should be noted, however, that due to the "repetition rule" in definition A.5, an equivalence-based dilemma derivation can go on forever without deriving any new consequences. Thus, for equivalence-based dilemma we lose the upper bound on length (and consequently also on size) of proper d-derivations given in theorem 4.23 on page 58 (though a similar bound could be recovered by taking more care in designing the dilemma rule (A.1) and/or restricting what kinds of formula equivalence repetitions are admissible in definition A.5).

Below we first discuss the difference between equivalence-based and formula relation dilemma and then give informal proofs for our claims about the polynomial relations between the two proof systems for derivation length and size.

A.5.1 Comparison of the Dilemma Variants

In formula relation dilemma derivations, instead of dealing with formula equivalences we operate directly on (sub)formula equivalence relations R. At each propagation step in such a derivation, we apply one of the propagation rules in figures A.2, A.3, A.4 or A.5 and then update the affected equivalence classes in the formula relation by taking the closure under transitivity and complement. This means that in the next step in a derivation, we automatically have access to all implicit formula equivalences which can be derived by the equivalence rules for commutativity (rule (E2)), transitivity (rules (E3), (E4) and (E5)), complement (rules (E6), (E7) and (E8)) and introduction (rules (E9) and (E10)).

We give a short example to clarify this. Suppose that we know before an application of a rule that $P \equiv Q$ and $R \equiv S$ and as a result derive $Q \equiv R$. Then in formula relation dilemma we may freely in all succeeding derivation steps use the equivalence $P \equiv S$ (indeed, P and S will be interchangeable representatives of the same equivalence class). In contrast, before using the equivalence $P \equiv S$ in equivalence-based dilemma we have to derive it explicitly by two applications of the transitivity rule (E3), deducing first that $P \equiv Q$ and $Q \equiv R$ imply $P \equiv R$, and then that $P \equiv R$ and $R \equiv S$ imply $P \equiv S$.

As to the dilemma rule, when merging the two branches of a formula relation dilemma rule application, we derive in one step the new formula relation $\mathsf{R}_1 \sqcap \mathsf{R}_2$. Instead of the rule for contradiction (E11) in equivalence-based dilemma we have the formula relation variant

$$\frac{P \equiv P^C}{Q \equiv \top} \quad \forall Q \in \mathit{Sub}\,(F) \tag{A.4}$$

(rule (4.18) on page 48) and we can summarize the three cases (A.1), (A.2) and (A.3) of the dilemma rule for equivalence-based dilemma in one dilemma rule for formula relation dilemma:

$$\frac{\dfrac{\mathsf{R}}{\begin{array}{cc} \mathsf{R}[P \equiv Q] & \mathsf{R}[P \equiv Q^C] \\ \pi_1 & \pi_2 \\ \mathsf{R}_1 & \mathsf{R}_2 \end{array}}}{\mathsf{R}_1 \sqcap \mathsf{R}_2} \tag{A.5}$$

(rule (4.19) on page 49). In contrast, in equivalence-based dilemma we have to restate explicitly the derived formula equivalences before and after closing a dilemma rule application.

A.5.2 Derivation Length

It follows from the above discussion that derivations in the equivalence-based dilemma proof system will in general be longer than corresponding derivations in formula relation dilemma. However, the difference in length is at most polynomial. We sketch a proof of this fact below.

We claim that for any derivation π in formula relation dilemma we can construct a corresponding equivalence-based dilemma derivation π' in such a way that the following holds: Suppose that we have translated the formula relation derivation π to a equivalence-based derivation π' up to a certain point, possibly somewhere inside one or more dilemma rule applications, and suppose that at this point in π we derive the formula relation R_j by a simple rule from R_i or by the dilemma rule from R_i and $\mathsf{R}_{i'}$. Then we can derive from the active equivalences in our equivalence-based dilemma derivation π' the formula equivalences needed to generate R_j from R_i (and possibly $\mathsf{R}_{i'}$) in a number of derivation steps at most quadratic in the number of preceding formula relations in π. Granted that this claim holds, we can construct for a formula relation dilemma derivation π of length L a corresponding derivation π' in equivalence-based dilemma of length at most $\mathrm{O}\left(\sum_{i=1}^{L} i^2\right) = \mathrm{O}\left(L^3\right)$, i.e. with length $L(\pi')$ at most cubic in $L(\pi)$.

To show that the claim is true, we have to look at two cases: propagation and dilemma rule application.

For propagation, assume that at a certain point in a formula relation dilemma derivation we derive $R_1 \equiv R_2$ from (implicit) equivalences $P_1 \equiv P_i$ and $Q_1 \equiv Q_j$. But the equivalences $P_1 \equiv P_i$ and $Q_1 \equiv Q_j$ are closures under transitivity and complement of chains of equivalences derived earlier in the derivation, which can be written (not necessarily in order of derivation and possibly after renumbering)

$$P_1 \equiv P_2,\ P_2 \equiv P_3,\ \ldots,\ P_{i-1} \equiv P_i \tag{A.6}$$

and

$$Q_1 \equiv Q_2,\ Q_2 \equiv Q_3,\ \ldots,\ Q_{j-1} \equiv Q_j. \tag{A.7}$$

We see that in order to transform the derivation step in formula relation dilemma above to a legal equivalence-based dilemma derivation, it is sufficient to go through these chains and for each equivalence $P_{k-1} \equiv P_k$ add a derivation $P_1 \equiv P_{k-1},\ P_{k-1} \equiv P_k \Rightarrow P_1 \equiv P_k$, using the equivalence rules for commutativity, transitivity, complement and introduction (and similarly for equivalences $Q_{k-1} \equiv Q_k$). Finally, we apply the needed propagation rule to $P_1 \equiv P_i$ and $Q_1 \equiv Q_j$ to derive $R_1 \equiv R_2$. In all, we shall have to insert no more than a constant number of derivation steps for each preceding relation in the formula relation dilemma derivation.

Transforming dilemma rule applications is slightly more complicated. Suppose that we have the formula relation derivation fragment in figure A.6, where π_f does not contain any closed dilemma rule applications but is the "path" of formula relations leading up to the first closed dilemma rule application in some branch of the derivation (and is thus not necessarily a dilemma derivation itself), and the derivations $\pi_1 : \mathsf{R}\big[P \equiv Q\big] \Rightarrow \mathsf{R}_1$ and $\pi_2 : \mathsf{R}\big[P \equiv Q^C\big] \Rightarrow \mathsf{R}_2$ in the branches do not contain any dilemma rule applications but only simple derivations. Suppose, furthermore, that the formula relation derivation fragment up to and including R has been translated to an equivalence-based derivation fragment π'_f as described above. To construct an equivalence-based dilemma rule application corresponding to figure A.6, we proceed as follows.

The branching and the two subderivations π_1 and π_2 do not present any problems. We insert the dilemma rule assumption $P \equiv Q$ in the left branch and $P \equiv Q^C$ in the right branch and then construct corresponding equivalence-based derivations π'_1 and π'_2 as described above for propagation.

$$
\begin{array}{cc}
\multicolumn{2}{c}{\pi_f} \\
\multicolumn{2}{c}{\mathsf{R}} \\
\hline
\mathsf{R}[P \equiv Q] & \mathsf{R}[P \equiv Q^C] \\
\pi_1 & \pi_2 \\
\mathsf{R}_1 & \mathsf{R}_2 \\
\hline
\multicolumn{2}{c}{\mathsf{R}_1 \sqcap \mathsf{R}_2}
\end{array}
$$

Figure A.6: Derivation fragment in formula relation dilemma.

$$
\begin{array}{cc}
\multicolumn{2}{c}{\pi'_f} \\
\hline
P \equiv Q & P \equiv Q^C \\
\pi'_1 & \pi'_2 \\
\pi''_1 & \pi''_2 \\
\psi_1 & \psi_1 \\
\vdots & \vdots \\
\psi_k & \psi_k \\
\hline
\multicolumn{2}{c}{\psi_1} \\
\multicolumn{2}{c}{\vdots} \\
\multicolumn{2}{c}{\psi_k}
\end{array}
$$

Figure A.7: Derivation fragment transformed to equivalence-based dilemma.

As to the merging of the two branches, let $\Psi = \{\psi_1, \ldots, \psi_k\}$ be a minimal set of formula associations needed to generate $\mathsf{R}_1 \sqcap \mathsf{R}_2$ from R. These are the equivalences $\psi_1, \ldots, \psi_k$ of the equivalence-based dilemma rule (A.1) on page 130 that we will need to derive explicitly in the two branches of the dilemma rule application in π' and then restate after merging the branches. It must certainly hold that $|\Psi| \leq \max\{L(\pi_1), L(\pi_2)\} \leq L(\pi_1) + L(\pi_2)$ (since R_1 and R_2 are generated from $L(\pi_1)$ and $L(\pi_2)$ number of formula equivalences respectively).

All associations in Ψ are consequences under closure by transitivity and complement of formula equivalences derived explicitly in π and π_1 on the one hand and π and π_2, on the other. Therefore, we can make equivalence-based derivations π''_1 and π''_2 in the two branches where each of the at most $L(\pi_1) + L(\pi_2)$ ψ_i's are derived by the equivalence rules for commutativity, transitivity, complement and introduction in length $\mathrm{O}(L(\pi) + L(\pi_1) + L(\pi_2))$. We then restate the equivalences ψ_i at the bottom of both branches, close the dilemma rule application and repeat the ψ_i once more.

The resulting derivation fragment in equivalence-based dilemma is presented in figure A.7, where

$$L(\pi'_f) = \mathrm{O}\left(L(\pi_f)^2\right) \tag{A.8}$$

$$L(\pi'_i) = \mathrm{O}\left((L(\pi_f) + L(\pi_1) + L(\pi_2))^2\right) \tag{A.9}$$

$$L(\pi''_i) = \mathrm{O}\left((L(\pi_f) + L(\pi_1) + L(\pi_2))^2\right) \tag{A.10}$$

$$|\{\psi_1, \ldots, \psi_k\}| = \mathrm{O}(L(\pi_1) + L(\pi_2)) \tag{A.11}$$

We see that the formula relation derivation fragment in figure A.6 can be transformed to a corresponding equivalence-based derivation fragment in figure A.7 with length at most quadratic in the number of formula relations in figure A.6. This special case can be generalized to the case where π_1 and π_2 are derivations with an arbitrary amount of nested dilemma rule applications and π_f is an arbitrary dilemma derivation fragment in the sense described above. We leave the details to the reader.

A.5.3 Derivation Size

Since the length of a minimal equivalence-based dilemma derivation is at most cubic in the length of the corresponding formula relation dilemma derivation, the same bound certainly holds for the sizes of the two derivations.

More than that, one can argue that the size of a minimal equivalence-based dilemma derivation is at most *quadratic* in the length of the corresponding formula relation derivation, since in the latter derivation the full formula relation (including the "equivalence chains" described above) is given at each step in the derivation. However, since a cubic bound is enough for our purposes, we choose not to elaborate on this any further.

Appendix B

Two Proofs

This appendix contains the missing parts in the proofs of two of the theorems in chapter 6. In section B.1 we prove the lower bound on hardness in atomic dilemma needed in theorem 6.1, and in section B.2 we fill in the details in the proof of theorem 6.20.

B.1 Separation of $\mathcal{D}_B$ from $\mathcal{D}_A$ w.r.t. Hardness

Theorem 6.1 says that there exists a linear separation of bivalent dilemma from atomic dilemma with respect to hardness. In section 6.1 we proved that the hardness in bivalent dilemma of the suggested separation formulas CM_n was constant, but only sketched a proof of the $\Omega(n)$-bound on the hardness of CM_n in atomic dilemma needed for the separation. We now formalize the argument given in the proof sketch for the lower bound.

First, we generalize the matrix formulas CM_n to n by m matrices. Recalling the definitions

$$R_i^m := \bigwedge_{j=1}^{m} x_{i,j} \tag{B.1}$$

$$C_j^n := \bigwedge_{i=1}^{n} \overline{x}_{i,j} \tag{B.2}$$

from figure 6.1, we define

$$CM_{n,m} := \bigvee_{i=1}^{n} R_i^m \ \wedge\ \bigvee_{j=1}^{m} C_j^n. \tag{B.3}$$

To save some typing, we introduce a special notation for the formula relation resulting from assuming $CM_{n,m}$ true and 0-saturating:

$$\mathsf{M}_{n,m} := Sat\big(CM_{n,m}{}^{\top}, 0\big). \tag{B.4}$$

The intuitive concept that was mentioned in the proof sketch of "deleting" the last column C_m^n is formalized as augmenting $\mathsf{M}_{n,m}$ by the set of associations

$$\Psi_{C_m^n} := \big\{R_1^{m-1} \equiv R_1^m \ldots, R_{n-1}^{m-1} \equiv R_{n-1}^m\big\} \tag{B.5}$$

and "deleting" the last row R_n^m corresponds to augmenting by

$$\Psi_{R_n^m} := \{C_1^{n-1} \equiv C_1^n \ldots, C_{m-1}^{n-1} \equiv C_{m-1}^n\}. \tag{B.6}$$

Next, we show a technical lemma.

Lemma B.1
Suppose that $\min\{n,m\} \geq 3$. *Then*

$$Sat\big(\mathsf{M}_{n,m}[x_{n,m} \equiv \top, \Psi_{C_m^n}], 0\big) \sqcap Sat\big(\mathsf{M}_{n,m}[x_{n,m} \equiv \bot, \Psi_{R_n^m}], 0\big) = \mathsf{M}_{n,m}.$$

Proof: Consider first $\mathsf{M}_{n,m} = Sat\big(CM_{n,m}{}^\top, 0\big)$. Assuming $CM_{n,m}$ true and propagating yields $\bigvee_{j=1}^n R_i^m \equiv \top$ and $\bigvee_{i=1}^m C_j^n \equiv \top$ by rule (C1$_{NF}$). No further consequences can be derived by the propagation rules in figures 5.1 and 5.2, so $\mathsf{M}_{n,m}$ consists of the *TRUE*-class

$$\Big[\top, CM_{n,m}, \bigvee_{i=1}^n R_i^m, \bigvee_{j=1}^m C_j^n,\Big] \tag{B.7}$$

and its complementing *FALSE*-class. Note that according to the definition of formula relations in NF-dilemma (equation (5.3) on page 85), the unit equivalence classes $[P]$ for all other subformulas P in $CM_{n,m}$ are not in $\mathsf{M}_{n,m}$.

If we augment $\mathsf{M}_{n,m}$ by $x_{n,m} \equiv \top$ and propagate, we derive for the "parent subformulas" of $x_{n,m}$ the equivalences

$$R_n^m \equiv R_n^{m-1} \qquad \text{[by rule (C4}_{NF}\text{)]} \tag{B.8}$$

$$C_m^n \equiv \bot \qquad \text{[by rule (C5}_{NF}\text{)]} \tag{B.9}$$

as well as the less interesting equivalences

$$R_n^m \setminus V_1 \equiv R_n^{m-1} \setminus V_1 \qquad \text{[by rule (C4}_{NF}\text{)]} \tag{B.10}$$

for all subsets $\emptyset \neq V_1 \subset \{x_{n,1}, \ldots x_{n,m-1}\}$ (i.e. for all subclauses of R_n^m containing $x_{n,m}$) and

$$C_m^n \setminus V_2 \equiv \bot \qquad \text{[by rule (C5}_{NF}\text{)]} \tag{B.11}$$

for all subsets $\emptyset \neq V_2 \subset \{\overline{x}_{1,m}, \ldots, \overline{x}_{n-1,m}\}$. This is all that can be derived by propagation on $x_{n,m} \equiv \top$. Using the new equivalences (B.8)–(B.11) we get

$$\bigvee_{j=1}^{m-1} C_j^n \equiv \bigvee_{j=1}^{m} C_j^n \qquad \text{[by (B.9) and rule (D5}_{NF}\text{)]} \tag{B.12}$$

as well as

$$\bigvee_{j\in S} C_j^n \equiv \bigvee_{j\in S} C_j^n \vee C_m^n \qquad \text{[by (B.9) and rule (D5}_{NF}\text{)]} \tag{B.13}$$

for all $\emptyset \neq S \subset [m-1]$. It is straightforward to check that nothing more can be derived nor about "parent subformulas" nor about "children subformulas" of the formulas in (B.8)–(B.13) by the propagation rules in figures 5.1 and 5.2.

Now add the associations $\{R_1^{m-1} \equiv R_1^m \ldots, R_{n-1}^{m-1} \equiv R_{n-1}^m\}$ in $\Psi_{C_m^n}$. The equivalence classes originating from $\Psi_{C_m^n}$ have no common elements with the classes arising as a result of the equivalences derived in (B.8)–(B.13), so any new associations resulting from the addition of $\Psi_{C_m^n}$ must be derived by application of the propagation rules for conjunction on the classes $[R_i^m, R_i^{m-1}]$, $i = 1, \ldots, n$. An inspection of figure 5.1 reveals that there are no applicable rules. That is, the formula relation is 0-saturated.

Summarizing the above, we can depict the relation

$$Sat\big(\mathsf{M}_{n,m}[x_{n,m} \equiv \top, \Psi_{C_m^n}], 0\big) \tag{B.14}$$

schematically as consisting of:

$\top$-1 the *TRUE*-class

$$\Big[\top, CM_{n,m}, \bigvee_{i=1}^{n} R_i^m, \bigvee_{j=1}^{m} C_j^n, \bigvee_{j=1}^{m-1} C_j^n, x_{n,m}, \neg C_m^n\Big] \\ \cup \big[\neg\big(C_m^n \setminus V_2\big) \mid \emptyset \neq V_2 \subset \{\overline{x}_{1,m}, \ldots, \overline{x}_{n-1,m}\}\big],$$

$\top$-2 the equivalence classes $[R_i^m, R_i^{m-1}]$ for $i = 1, \ldots, n$,

$\top$-3 the "subclasses" $\{[R_n^m \setminus V_1, R_n^{m-1} \setminus V_1] \mid \emptyset \neq V_1 \subset \{x_{n,1}, \ldots, x_{n,m-1}\}\}$ of the equivalence class $[R_n^m, R_n^{m-1}]$,

$\top$-4 the "subclasses" $\Big\{\big[\bigvee_{j\in S} C_j^n, \bigvee_{j\in S} C_j^n \vee C_m^n\big] \mid \emptyset \neq S \subset [m-1]\Big\}$ in (B.13) of (B.12)

and complementary equivalence classes. In an analogous fashion we derive that

$$Sat\big(\mathsf{M}_{n,m}[x_{n,m} \equiv \bot, \Psi_{R_n^m}], 0\big) \tag{B.15}$$

has the following equivalence classes (with complements):

$\bot$-1 the *TRUE*-class

$$\Big[\top, CM_{n,m}, \bigvee_{i=1}^{n} R_i^m, \bigvee_{j=1}^{m} C_j^n, \bigvee_{i=1}^{n-1} R_i^m, \overline{x}_{n,m}, \neg R_n^m\Big] \\ \cup \big[\neg\big(R_n^m \setminus V_1\big) \mid \emptyset \neq V_1 \subset \{x_{n,1}, \ldots, x_{n,m-1}\}\big],$$

$\bot$-2 the classes $[C_j^n, C_j^{n-1}]$, $j = 1, \ldots, m$,

$\bot$-3 the "subclasses" $\{[C_m^n \setminus V_2, C_m^{n-1} \setminus V_2] \mid \emptyset \neq V_2 \subset \{\overline{x}_{1,m}, \ldots, \overline{x}_{n-1,m}\}\}$,

$\bot$-4 the "subclasses" $\{[\bigvee_{i\in S} R_i^m, \bigvee_{i\in S} R_i^m \vee R_n^m] \mid \emptyset \neq S \subset [n-1]\}$.

It is now a routine matter to verify the statement of the lemma that the formula relation intersection of (B.14) and (B.15) is $\mathsf{M}_{n,m}$ by investigating the pairwise intersections of the equivalence classes in ($\top$-1)–($\top$-4) and ($\bot$-1)–($\bot$-4). We remind the reader that by the definition in (5.3), pairwise intersections that yield unit equivalence classes are not included in the domain of the formula relation intersection.

The intersection of the *TRUE*-classes ($\top$-1) and ($\perp$-1) is easily seen to be (B.7). The intersection of one *TRUE*- and one *FALSE*-class is the unit equivalence class $[x_{n,m}]$, which is ignored.

Intersections of classes in ($\top$-1)–($\top$-4) with classes in ($\perp$-2) yield only unit equivalence classes, since C_j^{n-1} is not in the domain of (B.14). This holds also for ($\top$-1)–($\top$-4) and ($\perp$-3), since no $C_m^{n-1} \setminus V_2$ is in the domain of (B.14). The intersections of ($\top$-1)–($\top$-4) with ($\perp$-4), finally, also result only in unit classes since $\bigvee_{j=1}^{n-1} R_i^m$ and $\bigvee_{i \in S} R_i^m$ (for $\emptyset \neq S \subset [n-1]$) are not in (B.14).

In the same way, we get that the intersections of classes in ($\perp$-1) with classes in ($\top$-2)–($\top$-4) yield nothing but unit equivalence classes. Thus, the formula relation intersection of (B.14) and (B.15) is $\mathsf{M}_{n,m}$, which was to be proved. $\square$

Using lemma B.1 and the strategy laid out in proposition 5.11, we can prove an $\Omega(\min\{n,m\})$-bound on any nontrivial atomic derivation from $\mathsf{M}_{n,m}$.

Lemma B.2
Suppose that $\min\{n,m\} \geq 2$ *and let* π *be a proper nontrivial atomic NF-dilemma derivation* $\pi : \mathsf{M}_{n,m} \Rightarrow \mathsf{M}'_{n,m}$. *Then*

$$D(\pi) \geq \min\{n,m\} - 1$$

and in particular in holds that

$$H_{\mathcal{D}_A}(\mathsf{M}_{n,m}) \geq \min\{n,m\} - 1.$$

Proof: We prove lemma B.2 by a somewhat modified variant of the proof of proposition 5.11 with induction over n and m.

By the proof of lemma B.1, the relation $\mathsf{M}_{n,m}$ is 1-hard for $\min\{n,m\} \geq 2$, which guarantees the existence of nontrivial derivations (this is condition 1 in proposition 5.11).

For $\min\{n,m\} \leq 2$ there is nothing to prove, since the inequality in the lemma is true by definition. Suppose that the lemma holds for $\mathsf{M}_{n,m-1}$ and $\mathsf{M}_{n-1,m}$ with $\min\{n,m\} \geq 3$, and let π be a proper nontrivial atomic derivation from $\mathsf{M}_{n,m}$.

Since $\mathsf{M}_{n,m}$ is 0-saturated, π must start by a dilemma rule application, without loss of generality branching over $x_{n,m}$. Then π can be written as

$$\begin{array}{c}
\mathsf{M}_{n,m} \\
\hline
\begin{array}{cc}
\mathsf{M}_{n,m}[x_{n,m} \equiv \top] & \mathsf{M}_{n,m}[x_{n,m} \equiv \perp] \\
\pi_1 & \pi_2 \\
\mathsf{R}_1 & \mathsf{R}_2
\end{array} \\
\hline
\mathsf{R}_1 \sqcap \mathsf{R}_2 \\
\pi_3 \\
\mathsf{M}'_{n,m}
\end{array} \tag{B.16}$$

where $\pi_1 : \mathsf{M}_{n,m}[x_{n,m} \equiv \top] \Rightarrow \mathsf{R}_1$ and $\pi_2 : \mathsf{M}_{n,m}[x_{n,m} \equiv \perp] \Rightarrow \mathsf{R}_2$ are subderivations such that

$$\mathsf{R}_1 \sqcap \mathsf{R}_2 \sqsupset \mathsf{M}_{n,m} \tag{B.17}$$

since π is proper. Note that we must have $\max\{D(\pi_1), D(\pi_2),\} \geq 1$ (this is condition 2 in proposition 5.11) since it follows from lemma B.1 that

$$Sat(\mathsf{M}_{n,m}[x_{n,m} \equiv \top], 0) \sqcap Sat(\mathsf{M}_{n,m}[x_{n,m} \equiv \perp], 0) = \mathsf{M}_{n,m}. \tag{B.18}$$

Now consider the augmented derivations

$$\pi_1\left[\Psi_{C_m^n}\right] : \mathsf{M}_{n,m}\left[x_{n,m} \equiv \top, \Psi_{C_m^n}\right] \Rightarrow \mathsf{R}_1\left[\Psi_{C_m^n}\right] \tag{B.19}$$

and

$$\pi_2\left[\Psi_{R_n^m}\right] : \mathsf{M}_{n,m}\left[x_{n,m} \equiv \bot, \Psi_{R_n^m}\right] \Rightarrow \mathsf{R}_2\left[\Psi_{R_n^m}\right]. \tag{B.20}$$

In view of lemma B.1, it must either be the case that

$$\mathsf{R}_1\left[\Psi_{C_m^n}\right] \sqsupset Sat\left(\mathsf{M}_{n,m}\left[x_{n,m} \equiv \top, \Psi_{C_m^n}\right], 0\right) \tag{B.21}$$

or that

$$\mathsf{R}_2\left[\Psi_{R_n^m}\right] \sqsupset Sat\left(\mathsf{M}_{n,m}\left[x_{n,m} \equiv \bot, \Psi_{R_n^m}\right], 0\right) \tag{B.22}$$

(otherwise $\mathsf{R}_1 \sqcap \mathsf{R}_2 \sqsubseteq \mathsf{M}_{n,m}$, which contradicts (B.17)).

Suppose that the inequality (B.21) holds. Then we can augment the derivation (B.19) with all associations in $Sat\left(\mathsf{M}_{n,m}\left[x_{n,m} \equiv \top, \Psi_{C_m^n}\right], 0\right)$ and remove redundant derivation steps to get a proper nontrivial derivation

$$\pi_1' : Sat\left(\mathsf{M}_{n,m}\left[x_{n,m} \equiv \bot, \Psi_{R_n^m}\right], 0\right) \Rightarrow \mathsf{R}_1'. \tag{B.23}$$

We now show that π_1' can be used to construct a proper nontrivial derivation π_1'' from $\mathsf{M}_{n,m-1}$ in depth $D\left(\pi_1''\right) \leq D\left(\pi_1\right)$ (this is condition 3 in proposition 5.11).

The derivation π_1' must start by a dilemma rule application. If the dilemma rule branches over $x_{i,j}$ for $j \leq m-1$, we let our constructed derivation π_1'' branch over the same variable. If the branch is over some variable $x_{i,m}$ in column m, we let π_1'' branch over $x_{i,m-1}$.

Now suppose π_1' applies some propagation rule which derives an equivalence $P_1 \equiv P_2$. If the subformulas P_i of $CM_{n,m}$ are also subformulas of $CM_{n,m-1}$, the derivation π_1'' can apply the same rule on the same subformulas to derive the same equivalence. If one or both of the P_i are formulas R_i^m or $R_i^m \setminus V_1$ containing $x_{i,m}$, then π_1'' can derive corresponding equivalences for $P_i' = R_i^{m-1}$ or $P_i' = R_i^{m-1} \setminus V_1$. Finally, if one or both of the P_i are formulas C_n^m or $C_n^m \setminus V_2$, then π_1'' can derive corresponding equivalences for $P_i' = C_n^{m-1}$ or $P_i' = C_n^{m-1} \setminus V_2'$ (where $\overline{x}_{i,m-1} \in V_2'$ if and only if $\overline{x}_{i,m} \in V_2$). This is so since branches over $x_{i,m}$ in π_1' correspond to branches over $x_{i,m-1}$ in π_1''.

The same reasoning can be applied to nested dilemma rule applications within the two branches in π_1'. In this way, as long as π_1' branches and propagates, the derivation π_1'' can mimic the derivation steps in π_1' as described above. It follows that if π_1' closes a dilemma rule application deriving common new associations in the two branches (since π_1' is proper), then π_1'', too, can close its corresponding dilemma rule application and derive common new associations.

Thus, since π_1' is a proper nontrivial derivation, so is π_1''. But if π_1'' is a proper nontrivial derivation from $\mathsf{M}_{n,m-1}$, we must have

$$D\left(\pi_1\right) \geq D\left(\pi_1'\right) \geq D\left(\pi_1''\right) \geq \min\left\{n, m-1\right\} - 1, \tag{B.24}$$

(where the last inequality follows by the induction hypotesis).

If (B.22) holds, we can derive the bound

$$D\left(\pi_2\right) \geq \min\left\{n-1, m\right\} - 1 \tag{B.25}$$

from the induction hypotesis for $\mathsf{M}_{n-1,m}$ in an analogous way.

Since either (B.24) or (B.25) must be true, we conclude that

$$D(\pi) \geq \min\{n, m\} - 1. \tag{B.26}$$

The lemma follows by the induction principle. □

It is an immediate consequence of lemma B.2 that the hardness degree of CM_n in atomic NF-dilemma is at least $n-1$, and by the reasoning in section 5.2.1, the same lower bound must hold in binary atomic dilemma for any parenthesized propositional logic version of the formula. (In fact, it is not hard to show that the hardness of CM_n in atomic dilemma is *exactly* $n-1$; see figure 6.4 on page 99.) This completes the proof of the linear separation in theorem 6.1.

B.2 Depth-Width Relation of $\mathcal{D}$ and $\mathcal{R}$

The results on dilemma and resolution presented in section 6.4 are based on theorem 6.20, which says that dilemma refutations π_D of CNF formulas F can be translated to resolution refutations π_R in width

$$W(\pi_R) = \mathrm{O}(W(F) \cdot D(\pi_D)) \tag{B.27}$$

and length

$$L(\pi_R) = \left(L(\pi_D) \cdot W(F)^{D(\pi_D)}\right)^{\mathrm{O}(1)}. \tag{B.28}$$

We never proved this theorem in section 6.4, however, but only sketched very roughly the intuition behind the proof. In this section, we fill in the missing details and give a formal proof of the theorem.

Suppose that F is an unsatisfiable CNF formula in width $W(F) = k$ and that π_D is an equivalence-based general NF-dilemma refutation of F. Without loss of generality, we may assume that π_D starts by deriving $C \equiv \top$ in depth 0 for all clauses $C \in F$ mentioned in the proof and that apart from this π_D does not contain any applications of the propagation rules for conjunction in figure 5.1. (If π_D does not meet these requirements it is easy to construct a new derivation π'_D from π_D in length $L(\pi'_D) \leq L(\pi_D)$ and width $W(\pi'_D) \leq W(\pi_D)$ that does).

As described in the proof sketch for theorem 6.20 in section 6.4, we transform π_D to a proof consisting of a sequence of lines by denoting the fact that the equivalence ϕ has been derived under (open) assumptions $\psi_1, \ldots, \psi_i$ from preceding lines in the proof by

$$\psi_1 \Rightarrow \ldots \Rightarrow \psi_i \Rightarrow \phi. \tag{B.29}$$

The first row in every new branch of π_D is not translated (i.e. the first row in the proof $F \equiv \top$ and the assumptions ψ and ψ^C in dilemma rule applications). When the proof π_D branches, we translate first the subderivation in the left branch of the dilemma rule application, then the right branch and finally the subderivation after the closure of the dilemma rule application (and recursively in the same way for the dilemma rule applications within the branches).

In this way, π_D is rewritten to a proof

$$\begin{array}{c} \psi_{1,1} \Rightarrow \ldots \Rightarrow \psi_{1,d_1} \Rightarrow \phi_1 \\ \vdots \\ \psi_{n,1} \Rightarrow \ldots \Rightarrow \psi_{n,d_n} \Rightarrow \phi_n \end{array} \tag{B.30}$$

in a somewhat more general proof system which we may call *line-based dilemma*. Lines in a line-based dilemma refutation can in turn be translated into sets of CNF clauses. These CNF clauses constitute the backbone of a resolution refutation of the formula F, the gaps in which can be completed in width and length as stated in theorem 6.20.

This section is organized as follows. We start by giving a formal definition of line-based dilemma in section B.2.1. In section B.2.2, we define the translation of lines in a line-based dilemma refutation to sets of CNF clauses and state theorem B.3 about translation from line-based dilemma to resolution, from which theorem 6.20 follows. The remaining part of the appendix is spent proving theorem B.3. After some preliminaries in section B.2.3, the proof boils down to a case analysis for the disjunction propagation rules in figure 5.2 in section B.2.4, the equivalence rules in figure A.1 in section B.2.5 and the dilemma rules (A.1), (A.2) and (A.3) in section B.2.6.

B.2.1 Formal Definition of Line-Based Dilemma

A *line-based dilemma derivation* from a CNF formula F is a sequence

$$\pi_L = \{H_1, \ldots, H_n\} \tag{B.31}$$

of *legal* lines

$$H_i = \psi_1 \Rightarrow \ldots \Rightarrow \psi_d \Rightarrow \phi \tag{B.32}$$

where ψ_j, ϕ are equivalences $P \equiv Q$ for P, Q unnegated or negated subclauses of the clauses of F.

In the rest of this appendix, we use the shorthand $\Psi \Rightarrow$ to denote a chain of assumptions $\psi_1 \Rightarrow \ldots \Rightarrow \psi_d \Rightarrow$. We write $\psi_j \in \Psi$ to denote that $\psi_j \Rightarrow$ is one of the assumptions in $\Psi \Rightarrow$. If all assumptions in $\Psi' \Rightarrow$ are found in $\Psi \Rightarrow$, we write $\Psi' \subseteq \Psi$. We use the notation $|\psi_1 \Rightarrow \ldots \Rightarrow \psi_d \Rightarrow| = d$ for the number of assumptions in a chain and define $|\Psi \Rightarrow \phi| := |\Psi \Rightarrow| + 1$ for a line-based dilemma derivation line.

A derivation line H_i in line-based dilemma is *legal* if it is derivable according to one of the following rules.

Axiom If C is a clause in F, then $H_i = C$ is a legal line.

Propagation The line $H_i = \Psi \Rightarrow \phi$ is a legal conclusion by propagation if there exists a disjunction propagation rule

$$\frac{\psi}{\phi}$$

in figure 5.2 on page 86 such that either $\psi \in \Psi$ or there is a line-based dilemma line $H_{i'} = \Psi' \Rightarrow \psi$ with $i' < i$ (i.e. occurring earlier in the derivation π_L) and $\Psi' \subseteq \Psi$.

Equivalence The line $H_i = \Psi \Rightarrow \phi$ is a legal conclusion by equivalence if there exists an equivalence relation rule

$$\frac{\psi_1, \ldots, \psi_c}{\phi}$$

in figure A.1 on page 126 such that for all $1 \leq j \leq c \leq 2$ either $\psi_j \in \Psi$ or there is a line-based dilemma line $H_{i_j} = \Psi'_j \Rightarrow \psi_j$ with $i_j < i$ and $\Psi'_j \subseteq \Psi$.

Dilemma $H_i = \Psi \Rightarrow \phi$ is a legal conclusion by the dilemma rule if there are line-based dilemma lines

$$H_{i_1} = \Psi \Rightarrow \psi' \Rightarrow \phi$$

and

$$H_{i_2} = \Psi \Rightarrow {\psi'}^C \Rightarrow \phi$$

or

$$H_{i_2} = \Psi \Rightarrow {\psi'}^C \Rightarrow \bot$$

with $i_1, i_2 < i$ (where we abuse notation slightly to allow for the special case $\phi = \bot$ in both branches).

The line-based dilemma derivation π_L is a refutation of F if it ends with the line $H_n = \bot$.

It is not hard to see that the translation of a a proper equivalence-based general NF-dilemma refutation π_D of F as described above yields a legal line-based dilemma refutation π_L of F in at most the same length and depth.

B.2.2 Translation of Line-Based Dilemma to CNF

We now define how the lines in a line-based dilemma derivation are translated to sets of CNF clauses. As described in section 6.4, each line

$$\psi_1 \Rightarrow \ldots \Rightarrow \psi_d \Rightarrow \phi \tag{B.33}$$

is translated to a set of CNF clauses $\mathit{CNF}(\psi_1 \Rightarrow \ldots \Rightarrow \psi_d \Rightarrow \phi)$ by interpreting (B.33) as

$$\neg\psi_1 \vee \ldots \vee \neg\psi_d \vee \phi \tag{B.34}$$

and rewriting this formula to conjunctive normal form. In the following, we will somewhat incorrectly denote a set of clauses $\{C \mid C \in \mathit{CNF}(\Psi \Rightarrow \phi)\}$ by their conjunction $\bigwedge_{C \in \mathit{CNF}(\Psi \Rightarrow \phi)} C$. We feel that this a convenient way to make the notation concise and avoid cluttering the proofs, and trust that it will not confuse the reader.

We start by defining translations to clauses in conjunctive normal form of single equivalences:

$$CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \top\right) := \bigvee_{i=1}^{n_1} a_i \tag{B.35}$$

$$CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \bot\right) := \bigwedge_{i=1}^{n_1} \overline{a_i} \tag{B.36}$$

$$\begin{aligned} CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j\right) &:= \bigwedge_{i=1}^{n_1} (\overline{a_i} \vee \bigvee_{j=1}^{n_2} b_j) \\ &\wedge \bigwedge_{j=1}^{n_2} (\overline{b_j} \vee \bigvee_{i=1}^{n_1} a_i) \end{aligned} \tag{B.37}$$

$$\begin{aligned} CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j\right) &:= \bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} (\overline{a_i} \vee \overline{b_j}) \\ &\wedge (\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j) \end{aligned} \tag{B.38}$$

The other types of single equivalences are reduced to the above cases in the obvious way:

$$CNF\left(\neg \bigvee_{i=1}^{n_1} a_i \equiv \top\right) := CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \bot\right) \tag{B.39}$$

$$CNF\left(\neg \bigvee_{i=1}^{n_1} a_i \equiv \bot\right) := CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \top\right) \tag{B.40}$$

$$CNF\left(\neg \bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j\right) := CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j\right) \tag{B.41}$$

$$CNF\left(\neg \bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j\right) := CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j\right) \tag{B.42}$$

Let ψ_i and ϕ denote arbitrary equivalences $P_i \equiv Q_i$ for P_i, Q_i unnegated or negated subclauses of the clauses of F and let $\Psi \Rightarrow$ denote a chain of assumptions $\psi_1 \Rightarrow \ldots \Rightarrow \psi_d \Rightarrow$. Then for line-based dilemma derivation lines we define:

$$CNF\,(\psi_i \Rightarrow) := CNF\left({\psi_i}^C\right) \tag{B.43}$$

$$CNF\,(\Psi \Rightarrow \psi_i \Rightarrow) := \bigwedge_{A \in CNF(\Psi \Rightarrow)} \bigwedge_{B \in CNF(\psi_i \Rightarrow)} (A \vee B) \tag{B.44}$$

$$CNF\,(\Psi \Rightarrow \bot) := CNF\,(\Psi \Rightarrow) \tag{B.45}$$

$$CNF\,(\Psi \Rightarrow \phi) := CNF\left(\Psi \Rightarrow \phi^C \Rightarrow \bot\right) \tag{B.46}$$

We adopt the convention $CNF\,(\emptyset \Rightarrow) = 0$ for an empty chain of assumptions $\emptyset \Rightarrow$ and $A \vee 0 = A \vee 0 = A$ for all clauses A (where 0 is the empty clause).

We are now ready to state formally theorem B.3, the proof of which is the goal of the rest of section B.2.

Theorem B.3 (Translation from line-based dilemma to resolution)
Let $\pi_L = \{H_1, \ldots, H_n\}$ be a line-based dilemma refutation (as defined in section B.2.1) of a CNF formula F with maximum clause width $W(F) = k$. Assume that π_L is proper *in the sense that no assumptions are made over clauses $C \in F$ and no equivalences $D_1 \equiv D_2$ or $D_1 \equiv \neg D_2$ are derived for clauses D_1, D_2 at a point in the derivation where instead $D_i \equiv \top/\bot$ can be derived in two simple derivation steps (in particular, this holds if $D_1, D_2 \in F$).*

Then for each line $H_i = \psi_{i,1} \Rightarrow \ldots \Rightarrow \psi_{i,d_i} \Rightarrow \phi_i$ in π_L, the set of CNF clauses

$$\mathit{CNF}\,(\psi_{i,1} \Rightarrow \ldots \Rightarrow \psi_{i,d_i} \Rightarrow \phi_i)$$

can be derived by the resolution and weakening rules from the sets of clauses

$$\begin{gathered}\mathit{CNF}\,(\psi_{1,1} \Rightarrow \ldots \Rightarrow \psi_{1,d_1} \Rightarrow \phi_1)\\ \vdots \\ \mathit{CNF}\,\bigl(\psi_{i-1,1} \Rightarrow \ldots \Rightarrow \psi_{i-1,d_{i-1}} \Rightarrow \phi_{i-1}\bigr)\end{gathered}$$

corresponding to preceding lines in the proof π_L in length at most $2 \cdot k^{2d+3}$ and width at most $(2d+3)(k-1)$ for $d = \max_{1 \le j \le i} \{d_j\}$.

In view of what has been said above, if we can prove theorem B.3 then theorem 6.20 follows immediately.

B.2.3 Some Auxiliary Lemmas and Observations

Before actually proving theorem B.3, we gather some facts which help us simplify and shorten the proof. Our first observation is an immediate consequence of the definitions in section B.2.2.

Observation B.4
If $H = \psi_1 \Rightarrow \ldots \Rightarrow \psi_d \Rightarrow \phi$ is a line-based dilemma line, then the translation of H to CNF clauses can be written as

$$\begin{aligned}&\mathit{CNF}\,(\psi_1 \Rightarrow \ldots \Rightarrow \psi_d \Rightarrow \phi) = \\ &\qquad = \bigwedge_{A_1 \in \mathit{CNF}(\psi_1 \Rightarrow)} \cdots \bigwedge_{A_d \in \mathit{CNF}(\psi_d \Rightarrow)} \bigwedge_{B \in \mathit{CNF}(\phi)} \Bigl(\bigvee_{i=1}^{d} A_i \vee B\Bigr).\end{aligned}$$

In particular, the translation to CNF is independent of the order of the assumptions in H.

This means that we can freely reorder the assumptions to simplify the proofs of derivability in resolution of the rules in line-based dilemma.

Lemma B.5
If π_L is a line-based dilemma derivation as described in theorem B.3, then for all lines $H_i = \psi_{i,1} \Rightarrow \ldots \Rightarrow \psi_{i,d_i} \Rightarrow \phi_i$ in π_L we have

$$L\,(\mathit{CNF}\,(\psi_{i,1} \Rightarrow \ldots \Rightarrow \psi_{i,d_i} \Rightarrow \phi_i)) \le k^{2d_i+2}$$

and

$$W\,(\mathit{CNF}\,(\psi_{i,1} \Rightarrow \ldots \Rightarrow \psi_{i,d_i} \Rightarrow \phi_i)) \le 2d_i(k-1) + (2k-1).$$

This follows from section B.2.2 and observation B.4. Note that by definition, the CNF clauses do not contain extra copies of any literals. Thus, we assume that for example a clause $a \vee b \vee b$ is implicitly reduced to $a \vee b$.

Often we can simplify the proofs by ignoring the set of assumptions under which the premises and the conclusion have been derived.

Observation B.6
Suppose that $H_i = \Psi \Rightarrow \phi$ was derived from lines $H_{i_j} = \Psi_j \Rightarrow \psi_j$, $j = 1, \ldots, c$ such that for all j it holds that $\Psi_j \subseteq \Psi$. Then to show that $CNF\,(\Psi \Rightarrow \phi)$ is derivable in resolution from the sets of clauses $CNF\,(\Psi_j \Rightarrow \psi_j)$, it suffices to show that $CNF\,(\phi)$ is derivable from $CNF\,(\psi_j)$, $j = 1, \ldots, c$.

For if so, we can first derive $CNF\,(\Psi \Rightarrow \psi_j)$ for all j by weakening and then insert $L\,(CNF\,(\Psi \Rightarrow))$ number of copies of the derivation of $CNF\,(\phi)$ from $CNF\,(\psi_j)$, $j = 1, \ldots, c$, one for each clause in $CNF\,(\Psi \Rightarrow)$.

Another simplifications is to throw away sets of clauses $CNF\,(\Psi \Rightarrow \phi)$ that are not needed in the proof. Clauses C that are not ordinary, i.e. that contain both x and $\overline{x}$ for some variable x, are tautological and can be omitted in a resolution refutation.

Observation B.7
If we delete all non-ordinary clauses in a resolution refutation π we get a valid refutation π' using the resolution and (possibly) weakening rules.

So if $CNF\,(\Psi \Rightarrow \phi)$ contains a subset S of non-ordinary clauses we can ignore both S and the derivation of S from preceding clauses. In particular:

Observation B.8
Clauses in sets of type $CNF\left(\Psi \Rightarrow \psi \Rightarrow \psi^C \Rightarrow \phi\right)$ and $CNF\left(\Psi \Rightarrow \psi \Rightarrow \psi^C \Rightarrow \bot\right)$ are non-ordinary and can be ignored in the proof of theorem B.3.

When we prove that the disjunction propagation rules and equivalence relation rules are derivable in resolution, we will run into a lot of non-ordinary clauses. The following lemma reduces the number of cases which need to be taken care of in the case analysis.

Lemma B.9
Suppose that ψ is an equivalence and that S is a set of clauses such that $\bigwedge_{C \in S} C$ is a subset of $CNF\,(\psi)$ or is derivable from $CNF\,(\psi)$ by weakening. Then

$$\bigwedge_{D \in CNF(\psi \Rightarrow)} \bigwedge_{C \in S} (C \vee D)$$

is a set of non-ordinary clauses.

Proof: If $\bigwedge_{C \in S} C$ is a subset of $CNF\,(\psi)$ or can be derived from $CNF\,(\psi)$ by weakening, every C must be a superset of some clause in $CNF\,(\psi)$. Now the lemma follows by an inspection of $\bigwedge_{D \in CNF(\psi \Rightarrow)} \bigwedge_{C \in S} \big(C \vee D\big)$ for the four cases for ψ in (B.35), (B.36), (B.37) and (B.38). □

We are now ready to prove that the bounds in theorem B.3 hold for all sets of clauses $CNF\,(\Psi \Rightarrow \phi)$ regardless of by which rule in line-based dilemma the line $H_i = \Psi \Rightarrow \phi$ was derived. The case with an axiom $H_i = C$, $C \in F$ is trivial, since by definition $CNF\,(C) = C$ is a legal conclusion in resolution if C is a clause in F. In the next three subsections, we attend to the other three kinds of rules for line-based dilemma derivations.

B.2.4 Derivability of Propagation Rules in Resolution

If the line $H_i = \Psi \Rightarrow \phi$ was concluded by a propagation rule for disjunction in figure 5.2 from some equivalence ψ we have two cases: Either $\psi \in \Psi$ or there is a line-based dilemma line $H_{i'} = \Psi' \Rightarrow \psi$ with $i' < i$ and $\Psi' \subseteq \Psi$.

For all propagation rules but one, we prove derivability in resolution by showing that if H_i was derived from $H_{i'}$ then $CNF\,(\Psi \Rightarrow \phi)$ can be obtained from $CNF\,(\Psi' \Rightarrow \psi)$ by weakening (and thus trivially satisfies the bounds on derivation length and width). For the case where $\psi \in \Psi$, it then follows by lemma B.9 that $CNF\,(\Psi \Rightarrow \phi)$ is a set of non-ordinary clauses.

Disjunction Propagation Rule ($\mathrm{D1}_{NF}$)

Rule ($\mathrm{D1}_{NF}$) derives $\bigvee_{i=1}^{n_1} a_i \equiv \bot$ from $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bot$.

Suppose that

$$H_i = \Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bot \tag{B.47}$$

was derived from an earlier line

$$H_{i'} = \Psi' \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bot. \tag{B.48}$$

By inspection of (B.36), we see that

$$CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \bot\right) \subseteq CNF\left(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bot\right), \tag{B.49}$$

and it follows (by observation B.6) that $CNF\big(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bot\big)$ is derivable from $CNF\big(\Psi' \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bot\big)$ by weakening.

If $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bot$ is one of the assumptions in H_i (say,

$$\Psi \Rightarrow \;=\; \Psi' \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bot \Rightarrow \tag{B.50}$$

without loss of generality because of observation B.4), then

$$CNF\left(\Psi' \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bot \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bot\right) \tag{B.51}$$

is a set of non-ordinary clauses by lemma B.9 (note that (B.49) shows that the preconditions of the lemma are satisfied).

Disjunction Propagation Rule $(\mathrm{D2}_{NF})$

Rule $(\mathrm{D2}_{NF})$ derives $\bigvee_{i=1}^{n_1} a_i \equiv \bot$ from $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i$.

Applying the definition in equation (B.38), we have

$$\begin{aligned} \mathit{CNF}\left(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i\right) = & \bigwedge_{i=1}^{n_1} \bigwedge_{i'=1}^{n_1} (\overline{a_i} \vee \overline{a_{i'}}) \\ \wedge & \bigwedge_{j=1}^{n_2} \bigwedge_{i=1}^{n_1} (\overline{b_j} \vee \overline{a_i}) \\ \wedge & \big(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j\big) \end{aligned} \tag{B.52}$$

which is a superset of $\mathit{CNF}\,(\bigvee_{i=1}^{n_1} a_i \equiv \bot)$ as defined by (B.36) (for $i = i'$ in $\bigwedge_{i=1}^{n_1} \bigwedge_{i'=1}^{n_1} (\overline{a_i} \vee \overline{a_{i'}})$).

So if $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i$ was derived by an earlier line, the set of clauses $\mathit{CNF}\,(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bot)$ is derivable from the clauses corresponding to that line by weakening. By lemma B.9, if $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i$ is one of the assumptions in Ψ the resulting set of CNF clauses is tautological.

Disjunction Propagation Rule $(\mathrm{D3}_{NF})$

Rule $(\mathrm{D3}_{NF})$ derives $\bigvee_{j=1}^{n_2} b_j \equiv \top$ from $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i$.

Note that we must have $n_1 + n_2 < k$. Otherwise, $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j$ is a clause in F and is known to be true, so $\bigvee_{j=1}^{n_2} b_j \equiv \top$ can be derived by applying rule $(\mathrm{D5}_{NF})$ and equivalence relation rules.

If $H_i = \Psi \Rightarrow \bigvee_{j=1}^{n_2} b_j \equiv \top$ was derived by an earlier line

$$H_{i'} = \Psi' \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i, \tag{B.53}$$

the set of clauses

$$\mathit{CNF}\left(\Psi \Rightarrow \bigvee_{j=1}^{n_2} b_j \equiv \top\right) = \bigwedge_{A \in \mathit{CNF}(\Psi \Rightarrow)} \big(\bigvee_{j=1}^{n_2} b_j \vee A\big) \tag{B.54}$$

can be derived by resolution over a_i, $i = 1, \ldots, n_1$ from clauses

$$\bigwedge_{A \in \mathit{CNF}(\Psi' \Rightarrow)} \bigwedge_{i=1}^{n_1} (\overline{a_i} \vee A) \tag{B.55}$$

and

$$\bigwedge_{A \in \mathit{CNF}(\Psi' \Rightarrow)} \big(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \vee A\big) \tag{B.56}$$

in the CNF translation of (B.53) (which is (B.52) complemented with CNF encodings of the assumptions Ψ) in length at most $k \cdot L\big(\mathit{CNF}\big(\Psi \Rightarrow \bigvee_{j=1}^{n_2} b_j \equiv \top\big)\big)$ and width no more than the widest CNF clause in the translations of H_i and $H_{i'}$.

Suppose instead that $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i$ is an assumption. If we ignore non-ordinary clauses, we have

$$
\begin{aligned}
CNF\left(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i \Rightarrow\right) \\
&= CNF\left(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{i=1}^{n_1} a_i\right) \\
&= \bigwedge_{j=1}^{n_2} (\overline{b_j} \vee \bigvee_{i=1}^{n_1} a_i) \qquad \text{(B.57)} \\
&= CNF\left(\bigvee_{j=1}^{n_2} b_j \equiv \top \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \top\right) \\
&= CNF\left(\bigvee_{j=1}^{n_2} b_j \equiv \top \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bot \Rightarrow\right).
\end{aligned}
$$

Using this equality and observation B.8, we get that

$$
\begin{aligned}
CNF\left(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{i=1}^{n_1} a_i \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bot\right) = \\
= CNF\left(\Psi \Rightarrow \bigvee_{j=1}^{n_2} b_j \equiv \top \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bot \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \top \Rightarrow \bot\right) \qquad \text{(B.58)}
\end{aligned}
$$

is a set of non-ordinary clauses.

Disjunction Propagation Rule (D4$_{NF}$)

Rule (D4$_{NF}$) derives $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \top$ from $\bigvee_{i=1}^{n_1} a_i \equiv \top$.

It follows immediately from the definition in equation (B.35) and lemma B.9 that $CNF\big(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \top\big)$ either can be derived by weakening or is a set of non-ordinary clauses.

Disjunction Propagation Rule (D5$_{NF}$)

Rule (D5$_{NF}$) derives $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{j=1}^{n_2} b_j$ from $\bigvee_{i=1}^{n_1} a_i \equiv \bot$.

From (B.57) we have the equality

$$
CNF\left(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{j=1}^{n_2} b_j\right) = \bigwedge_{i=1}^{n_1} (\overline{a_i} \vee \bigvee_{j=1}^{n_2} b_j), \qquad \text{(B.59)}
$$

which shows that either $CNF\big(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{j=1}^{n_2} b_j\big)$ can be derived by weakening or it is a set of non-ordinary clauses.

Disjunction Propagation Rule $(\mathrm{D}6_{NF})$

Rule $(\mathrm{D}6_{NF})$ derives $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{i=1}^{n_1} a_i$ from $\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j$.

Referring again to (B.57), we have

$$\begin{aligned} CNF\left(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{i=1}^{n_1} a_i\right) &= \bigwedge_{j=1}^{n_2} (\overline{b_j} \vee \bigvee_{i=1}^{n_1} a_i) \\ &\subseteq CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j\right) \end{aligned} \tag{B.60}$$

and it follows that $CNF\big(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{i=1}^{n_1} a_i\big)$ either can be derived by weakening or is a set of non-ordinary clauses.

Disjunction Propagation Rule $(\mathrm{D}7_{NF})$

Rule $(\mathrm{D}7_{NF})$ derives $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \top$ from $\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j$.

By the definitions in (B.35) and (B.38) it holds that

$$CNF\left(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \top\right) \subseteq CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j\right), \tag{B.61}$$

from which it follows that $CNF\big(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \equiv \top\big)$ can be derived by weakening or is a set of non-ordinary clauses.

Summary of Results for Disjunction Propagation Rules

If $H_i = \Psi \Rightarrow \phi$ is a line in a line-based dilemma derivation π_L where ϕ was derived by a disjunction propagation rule from a premise $\phi \in \Psi$, then $CNF\,(\Psi \Rightarrow \phi)$ is a set of non-ordinary clauses and can be ignored when translating π_L to a resolution derivation.

If the line $H_i = \Psi \Rightarrow \phi$ is the result of applying a disjunction propagation rule on an earlier line $H_{i'} = \Psi' \Rightarrow \psi$ in π_L, then for all rules except $(\mathrm{D}3_{NF})$ the set of clauses $CNF\,(\Psi \Rightarrow \phi)$ can be derived by weakening. For rule $(\mathrm{D}3_{NF})$, the translation to CNF of H_i can be derived in length at most

$$k \cdot L\,(CNF\,(\Psi \Rightarrow \phi)) \,\leq\, k^{|\Psi \Rightarrow \phi|+1} \tag{B.62}$$

and width at most

$$\max\left\{W\,(CNF\,(\Psi \Rightarrow \phi)), W\,(CNF\,(\Psi' \Rightarrow \psi))\right\} \,\leq\, 2 \cdot |\Psi \Rightarrow \phi| \cdot (k-1) \tag{B.63}$$

(where the last inequality holds since $\Psi' \subseteq \Psi$). Thus, the bounds in theorem B.3 hold for disjunction propagation rules.

B.2.5 Derivability of Equivalence Rules in Resolution

It is interesting to note that from the point of view of resolution, almost all dilemma derivation conclusions by propagation just restate self-evident facts either in the form of tautological sets of clauses (if the premise of the propagation rule is an assumption in the current line-based dilemma line) or as weakening of preceding clauses (if the premise was derived in an earlier dilemma derivation step). Most of the equivalence relation rules force resolution to work a bit harder, as we shall see, although rules for reflexivity and commutativity need not be translated at all for natural reasons.

Equivalence Relation Rule (E1)

The axiom $P \equiv P$ translates to a set $\mathit{CNF}\,(P \equiv P)$ of non-ordinary clauses and can thus be ignored.

Equivalence Relation Rule (E2)

Rule (E2) derives $Q \equiv P$ from $P \equiv Q$. Such derivation steps can be ignored in resolution since the two equivalences translates to the same set of CNF clauses.

Equivalence Relation Rule (E3)

Rule (E3) derives $Q \equiv R$ from $P \equiv Q$ and $Q \equiv R$.

Without loss of generality we may assume that the subformula P is an unnegated CNF clause $\bigvee_{i=1}^{n_1} a_i$ (otherwise derive $Q^C \equiv R^C$ from $P^C \equiv Q^C$ and $Q^C \equiv R^C$). For Q and R we then have three cases:

1. $Q = \bigvee_{j=1}^{n_2} b_j$ and $R = \bigvee_{l=1}^{n_3} c_l$,
2. $Q = \bigvee_{j=1}^{n_2} b_j$ and $R = \neg \bigvee_{l=1}^{n_3} c_l$,
3. $Q = \neg \bigvee_{j=1}^{n_2} b_j$ and $R = \bigvee_{l=1}^{n_3} c_l$.

(If $Q = \neg \bigvee_{j=1}^{n_2} b_j$ and $R = \neg \bigvee_{l=1}^{n_3} c_l$, we set $P' = R^C$, $Q' = Q^C$ and $R' = P^C$ and use the results for the second case above.) For all three cases, we get subcases depending on whether the premises are assumptions in the current line or have been derived in preceding lines.

In what follows, we will make frequent use of observation B.4, which makes it possible to to reorder the assumptions as wee see fit, and observation B.6, which allows us to ignore assumptions other than the premises $P \equiv Q$ and $Q \equiv R$.

Case 1: $P = \bigvee_{i=1}^{n_1} a_i$, $Q = \bigvee_{j=1}^{n_2} b_j$, $R = \bigvee_{l=1}^{n_3} c_l$.

(a) Both premises $\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j$ and $\bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{l=1}^{n_3} c_l$ are derived.

Consider

$$\mathit{CNF}\left(\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j\right) = \bigwedge_{i=1}^{n_1} (\overline{a_i} \vee \bigvee_{j=1}^{n_2} b_j) \wedge \bigwedge_{j=1}^{n_2} (\overline{b_j} \vee \bigvee_{i=1}^{n_1} a_i) \quad \text{(B.64)}$$

and

$$\mathit{CNF}\left(\bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{l=1}^{n_3} c_l\right) = \bigwedge_{j=1}^{n_2} (\overline{b_j} \vee \bigvee_{l=1}^{n_3} c_l) \wedge \bigwedge_{l=1}^{n_3} (\overline{c_l} \vee \bigvee_{j=1}^{n_2} b_j). \quad \text{(B.65)}$$

By resolving $\overline{a_i} \vee \bigvee_{j=1}^{n_2} b_j$ with $\overline{b_j} \vee \bigvee_{l=1}^{n_3} c_l$ for $j = 1, \ldots, n_2$, the set of clauses $\bigwedge_{i=1}^{n_1} \big(\overline{a_i} \vee \bigvee_{l=1}^{n_3} c_l\big)$ can be derived in $n_1 n_2$ steps in width $n_2 + n_3$. In the same way, $\bigwedge_{l=1}^{n_3} \big(\overline{c_l} \vee \bigvee_{i=1}^{n_1} a_i\big)$ can be derived in length $n_2 n_3$ and width $n_1 + n_2$. It follows that

$$CNF\left(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{l=1}^{n_3} c_l\right) =$$

$$\bigwedge_{A \in CNF(\Psi \Rightarrow)} \big(\bigwedge_{i=1}^{n_1} (\overline{a_i} \vee \bigvee_{l=1}^{n_3} c_l \vee A) \wedge \bigwedge_{l=1}^{n_3} (\overline{c_l} \vee \bigvee_{i=1}^{n_1} a_i \vee A)\big) \tag{B.66}$$

can be derived in length at most

$$\big(n_1 n_2 + n_2 n_3\big) \cdot L\left(CNF\left(\Psi \Rightarrow\right)\right) \leq 2 \cdot k^{2|\Psi \Rightarrow P \equiv R|} \tag{B.67}$$

and width at most

$$\max\left\{n_1 + n_2, n_2 + n_3\right\} + W\left(CNF\left(\Psi \Rightarrow\right)\right) \leq$$
$$\leq \big(2k - 1\big) + 2\big(k - 1\big)|\Psi \Rightarrow|. \tag{B.68}$$

(b) One of the premises is assumed, say $\bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{l=1}^{n_3} c_l$ without loss of generality because of symmetry.
Then

$$CNF\left(\bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{l=1}^{n_3} c_l \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{l=1}^{n_3} c_l\right)$$

$$= \bigwedge_{j=1}^{n_2} \bigwedge_{l=1}^{n_3} \bigwedge_{i'=1}^{n_1} \big(\overline{b_j} \vee \overline{c_l} \vee \overline{a_{i'}} \vee \bigvee_{l'=1}^{n_3} c_{l'}\big) \tag{B.69}$$

$$\wedge \bigwedge_{j=1}^{n_2} \bigwedge_{l=1}^{n_3} \bigwedge_{l'=1}^{n_3} \big(\overline{b_j} \vee \overline{c_l} \vee \overline{c_{l'}} \vee \bigvee_{i'=1}^{n_1} a_{i'}\big) \tag{B.70}$$

$$\wedge \bigwedge_{i'=1}^{n_1} \big(\bigvee_{j=1}^{n_2} b_j \vee \bigvee_{l=1}^{n_3} c_l \vee \overline{a_{i'}} \vee \bigvee_{l'=1}^{n_3} c_{l'}\big) \tag{B.71}$$

$$\wedge \bigwedge_{l'=1}^{n_3} \big(\bigvee_{j=1}^{n_2} b_j \vee \bigvee_{l=1}^{n_3} c_l \vee \overline{c_{l'}} \vee \bigvee_{i'=1}^{n_1} a_{i'}\big), \tag{B.72}$$

where (B.69) and (B.72) are non-ordinary and can be ignored and (B.70) and (B.71) follow from $CNF\big(\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j\big)$ by weakening.

(c) Both premises $\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j$ and $\bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{l=1}^{n_3} c_l$ are assumed.
By inspecting the calculations in case (1b) and combining the results with lemma B.9, we see that in this case the translation to conjunctive normal form yields a set of non-ordinary clauses which can be ignored.

Case 2: $P = \bigvee_{i=1}^{n_1} a_i$, $Q = \bigvee_{j=1}^{n_2} b_j$, $R = \neg \bigvee_{l=1}^{n_3} c_l$.

(a) Both premises $\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j$ and $\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l$ are derived.

Using (B.64) and

$$CNF\left(\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l\right) = \bigwedge_{j=1}^{n_2}\bigwedge_{l=1}^{n_3}(\overline{b_j} \vee \overline{c_l}) \wedge (\bigvee_{j=1}^{n_2} b_j \vee \bigvee_{l=1}^{n_3} c_l), \tag{B.73}$$

we can derive $\bigwedge_{i=1}^{n_1}\bigwedge_{l=1}^{n_3}(\overline{a_i} \vee \overline{c_l})$ by resolving $\overline{a_i} \vee \bigvee_{j=1}^{n_2} b_j$ with $\overline{b_j} \vee \overline{c_l}$ for $j = 1, \ldots, n_2$ for all i and l in a total of $n_1 n_2 n_3$ steps and in no extra width. Resolving $\bigvee_{j=1}^{n_2} b_j \vee \bigvee_{l=1}^{n_3} c_l$ with $\overline{b_j} \vee \bigvee_{i=1}^{n_1} a_i$ for $j = 1, \ldots, n_2$, we derive $\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{l=1}^{n_3} c_l$ in length n_2 and width $n_1 + n_2 + n_3 - 1$. Putting this together, we see that

$$CNF\left(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{l=1}^{n_3} c_l\right) =$$
$$\bigwedge_{A \in CNF(\Psi\Rightarrow)} (\bigwedge_{i=1}^{n_1}\bigwedge_{l=1}^{n_3}(\overline{a_i} \vee \overline{c_l} \vee A) \wedge (\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{l=1}^{n_3} c_l \vee A)) \tag{B.74}$$

can be derived in length at most

$$\big(n_1 n_2 n_3 + n_2\big) \cdot L\,(CNF\,(\Psi \Rightarrow)) \;\leq\; 2 \cdot k^{2|\Psi\Rightarrow P\equiv R|+1} \tag{B.75}$$

and width at most

$$n_1 + n_2 + n_3 - 1 + W\,(CNF\,(\Psi \Rightarrow)) \;\leq\; \leq\; \big(2|\Psi \Rightarrow| + 3\big)\big(k-1\big). \tag{B.76}$$

(b) $\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j$ is derived, $\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l$ is assumed.

Since

$$CNF\left(\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{l=1}^{n_3} c_l\right) = $$
$$= CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{l=1}^{n_3} c_l \Rightarrow \bigvee_{j=1}^{n_2} b_j \equiv \bigvee_{l=1}^{n_3} c_l\right), \tag{B.77}$$

this case reduces to case (1b).

(c) $\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l$ is derived, $\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j$ is assumed. We have

$$\begin{aligned} & CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{l=1}^{n_3} c_l\right) \\ = \ & \bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} \bigwedge_{i'=1}^{n_1} \bigwedge_{l'=1}^{n_3} (\overline{a_i} \vee \overline{b_j} \vee \overline{a_{i'}} \vee \overline{c_{l'}}) & \text{(B.78)} \\ \wedge \ & \bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} (\overline{a_i} \vee \overline{b_j} \vee \bigvee_{i'=1}^{n_1} a_{i'} \vee \bigvee_{l'=1}^{n_3} c_{l'}) & \text{(B.79)} \\ \wedge \ & \bigwedge_{i'=1}^{n_1} \bigwedge_{l'=1}^{n_3} (\bigvee_{i'=1}^{n_1} a_{i'} \vee \bigvee_{j=1}^{n_2} b_j \vee \overline{a_{i'}} \vee \overline{c_{l'}}) & \text{(B.80)} \\ \wedge \ & (\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \vee \bigvee_{i'=1}^{n_1} a_{i'} \vee \bigvee_{l'=1}^{n_3} c_{l'}). & \text{(B.81)} \end{aligned}$$

The clauses (B.79) and (B.80) are non-ordinary and can be ignored. The ordinary clauses in (B.78) and (B.81) follow from $CNF\left(\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l\right)$ by weakening (see (B.73)).

(d) Both premises $\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j$ and $\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l$ are assumed.

If both premises are assumptions, the translation to CNF results in a set of non-ordinary clauses by lemma B.9 applied on the calculations in case (2b) or case (2c).

Case 3: $P = \bigvee_{i=1}^{n_1} a_i$, $Q = \neg \bigvee_{j=1}^{n_2} b_j$, $R = \bigvee_{l=1}^{n_3} c_l$.

(a) Both premises $\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j$ and $\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l$ are derived.

Fix i and derive $\overline{a_i} \vee \bigvee_{l=1}^{n_3} c_l$ from $\overline{a_i} \vee \overline{b_i}$ and $\bigvee_{j=1}^{n_2} b_j \vee \bigvee_{l=1}^{n_3} c_l$ in n_2 steps (see (B.38) and (B.73)). In this way we get

$$\bigwedge_{A \in CNF(\Psi \Rightarrow)} \bigwedge_{i=1}^{n_1} (\overline{a_i} \vee \bigvee_{l=1}^{n_3} c_l \vee A) \tag{B.82}$$

in length $n_1 n_2 \cdot L\,(CNF\,(\Psi \Rightarrow))$ and no extra width. The complementary set of clauses

$$\bigwedge_{A \in CNF(\Psi \Rightarrow)} \bigwedge_{l=1}^{n_3} (\overline{c_l} \vee \bigvee_{i=1}^{n_1} a_i \vee A) \tag{B.83}$$

is derived in the same way for a total length of

$$\big(n_1 n_2 + n_2 n_3\big) \cdot L\,(CNF\,(\Psi \Rightarrow)) \ \le \ 2 \cdot k^{2|\Psi \Rightarrow P \equiv R|} \tag{B.84}$$

and width of

$$\begin{aligned} & \max\{n_1 + n_2, n_2 + n_3\} + W\,(CNF\,(\Psi \Rightarrow)) \ \le \\ & \qquad \le \ \big(2k - 1\big) + 2\big(k - 1\big)|\Psi \Rightarrow|. \end{aligned} \tag{B.85}$$

(b) One of the premises is assumed, say $\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l$ without loss of generality because of symmetry.

We leave it to the reader to verify that the (ordinary) clauses in $CNF\big(\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{l=1}^{n_3} c_l\big)$ are derivable by weakening from $CNF\big(\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j\big)$.

(c) Both premises $\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j$ and $\bigvee_{j=1}^{n_2} b_j \equiv \neg \bigvee_{l=1}^{n_3} c_l$ are assumed.

The non-ordinarity of the CNF translation follows from case (3b) by lemma B.9.

Equivalence Relation Rules (E4) **and** (E5)

The rules (E4) and (E5) are special cases of rule (E3), and the proof for the latter rule can easily be adapted to proofs for the former with the same bounds on length and width. We leave the details to the reader.

Equivalence Relation Rules (E6), (E7) **and** (E8)

The premises and conclusions in the equivalence rules (E6), (E7) and (E8) (which derive $P^C \equiv Q^C$ from $P \equiv Q$, $P^C \equiv \bot$ from $P \equiv \top$ and $P^C \equiv \top$ from $P \equiv \bot$, respectively) translate into identical sets of CNF clauses. These rules can therefore be ignored.

Equivalence Relation Rule (E9)

Rule (E9) derives $P \equiv Q$ from $P \equiv \top$ and $Q \equiv \top$.

Without loss of generality we may assume that the subformula P is an unnegated CNF clause $\bigvee_{i=1}^{n_1} a_i$ and that $Q = \bigvee_{j=1}^{n_2} b_j$ or $Q = \neg \bigvee_{j=1}^{n_2} b_j$ (if $P = \neg \bigvee_{i=1}^{n_1} a_i$, set $P' = P^C$ and $Q' = Q'^C$ and apply rule (E10)).

Case 1: $P = \bigvee_{i=1}^{n_1} a_i$, $Q = \bigvee_{j=1}^{n_2} b_j$.

If $\bigvee_{i=1}^{n_1} a_i \equiv \top$ and $\bigvee_{j=1}^{n_2} b_j \equiv \top$ has both been derived, it is easy to verify that $CNF\big(\bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j\big)$ follows from $CNF\big(\bigvee_{i=1}^{n_1} a_i \equiv \top\big)$ and $CNF\big(\bigvee_{j=1}^{n_2} b_j \equiv \top\big)$ by weakening.

If one of the premises, say $\bigvee_{j=1}^{n_2} b_j \equiv \top$, is an assumption, it is just as easy to check that $CNF\big(\bigvee_{j=1}^{n_2} b_j \equiv \top \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j\big)$ follows by weakening from $CNF\big(\bigvee_{i=1}^{n_1} a_i \equiv \top\big)$.

The non-ordinarity of the CNF translation in the case where both premises are assumptions now follows from lemma B.9.

Case 2: $P = \bigvee_{i=1}^{n_1} a_i$, $Q = \neg \bigvee_{j=1}^{n_2} b_j$.

We get four cases depending on whether $\bigvee_{i=1}^{n_1} a_i \equiv \top$ and $\neg \bigvee_{j=1}^{n_2} b_j \equiv \top$ are derived or assumed, but other than that the calculations (and results) are wholly analogous to case 1. We leave the details to the reader.

Equivalence Relation Rule (E10)

Rule (E10) derives $P \equiv Q$ from $P \equiv \bot$ and $Q \equiv \bot$.

Assume without loss of generality $P = \bigvee_{i=1}^{n_1} a_i$ and $Q = \bigvee_{j=1}^{n_2} b_j$ (the other cases can be reduced to rule (E9)). The case analysis and the calculations are wholly analogous to case 1 in rule (E9). We omit the details.

Equivalence Relation Rule (E11)

Rule (E11) derives $\bot$ from $P \equiv P^C$.

Assume $P = \bigvee_{i=1}^{n_1} a_i$ and note that $\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{i=1}^{n_1} a_i$ never can be an assumption but always is derived. The empty clause can be derived from

$$\begin{aligned} CNF\left(\bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{i=1}^{n_1} a_i\right) &= \bigwedge_{i=1}^{n_1} \bigwedge_{i'=1}^{n_1} (\overline{a_i} \vee \overline{a_{i'}}) \wedge \Big(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{i'=1}^{n_1} a_{i'}\Big) \\ &= \bigwedge_{i=1}^{n_1} \overline{a_i} \wedge \bigwedge_{i=1}^{n_1} \bigwedge_{i'=i+1}^{n_1} (\overline{a_i} \vee \overline{a_{i'}}) \wedge \Big(\bigvee_{i=1}^{n_1} a_i\Big) \end{aligned} \tag{B.86}$$

in n_1 steps and no extra width by resolving $\bigvee_{i=1}^{n_1} a_i$ with $\overline{a_i}$, $i = 1, \ldots, n_1$, so

$$CNF\,(\Psi \Rightarrow \bot) \tag{B.87}$$

is derivable from

$$CNF\left(\Psi' \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{i=1}^{n_1} a_i\right) \tag{B.88}$$

in length at most

$$n_1 \cdot L\,(CNF\,(\Psi \Rightarrow)) \;\leq\; 2 \cdot k^{2|\Psi \Rightarrow|+1} \tag{B.89}$$

and width no more than

$$\big(2|\Psi \Rightarrow| + 2\big)\big(k-1\big). \tag{B.90}$$

Summary of Results for Equivalence Relation Rules

Suppose that $H_i = \Psi \Rightarrow \phi$ is a line in a line-based dilemma derivation π_L derived by an equivalence relation rule (where we abuse notation by allowing $\phi = \bot$). Then the set of clauses $CNF\,(\Psi \Rightarrow \phi)$ can be derived by the resolution and weakening rules from CNF clauses corresponding to earlier lines in the proof π_L in length at most

$$2k^3 \cdot L\,(CNF\,(\Psi \Rightarrow)) \;\leq\; 2k^{|\Psi \Rightarrow|+3} \tag{B.91}$$

and width at most

$$3\big(k-1\big) + W\,(CNF\,(\Psi \Rightarrow)) \;\leq \big(2 \cdot |\Psi \Rightarrow| + 3\big)\big(k-1\big), \tag{B.92}$$

so the bounds in theorem B.3 hold also for equivalence relation rules.

B.2.6 Derivability of Dilemma Rule in Resolution

Suppose that the line $H_i = \Psi \Rightarrow \phi$ has been derived by the dilemma rule. Then by definition there are lines

$$H_{i_1} = \Psi \Rightarrow \psi' \Rightarrow \phi \tag{B.93}$$

and

$$H_{i_2} = \Psi \Rightarrow {\psi'}^C \Rightarrow \phi \tag{B.94}$$

or

$$H_{i_2} = \Psi \Rightarrow {\psi'}^C \Rightarrow \perp \tag{B.95}$$

with $i_1, i_2 < i$.

Assume first the most general case that $\psi' = \bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j$ and that H_{i_2} is on the form (B.94) (with $\phi \neq \perp$). Then we have

$$\begin{aligned} &\mathit{CNF}\left(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \bigvee_{j=1}^{n_2} b_j \Rightarrow \phi\right) = \\ &\quad = \bigwedge_{A \in \mathit{CNF}(\Psi \Rightarrow \phi)} \big(\bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} (\overline{a_i} \vee \overline{b_j} \vee A) \wedge \big(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \vee A\big)\big) \end{aligned} \tag{B.96}$$

and

$$\begin{aligned} &\mathit{CNF}\left(\Psi \Rightarrow \bigvee_{i=1}^{n_1} a_i \equiv \neg \bigvee_{j=1}^{n_2} b_j \Rightarrow \phi\right) = \\ &\quad = \bigwedge_{A \in \mathit{CNF}(\Psi \Rightarrow \phi)} \big(\bigwedge_{i=1}^{n_1} (\overline{a_i} \vee \bigvee_{j=1}^{n_2} b_j \vee A) \wedge \bigwedge_{j=1}^{n_2} (\overline{b_j} \vee \bigvee_{i=1}^{n_1} a_i \vee A)\big). \end{aligned} \tag{B.97}$$

For every $A \in \mathit{CNF}\,(\Psi \Rightarrow \phi)$, we derive $\overline{a_i} \vee A$ by resolution in n_2 steps from clauses $\overline{a_i} \vee \bigvee_{j=1}^{n_2} b_j \vee A$ in (B.97) and $\overline{a_i} \vee \overline{b_j} \vee A$, $j = 1, \ldots, n_2$ in (B.96). In the same way, we derive clauses $\overline{b_j} \vee A$. In a total of $2n_1 n_2 \cdot L\big(\mathit{CNF}\big(\Psi \Rightarrow \phi\big)\big)$ steps we get

$$\bigwedge_{A \in \mathit{CNF}(\Psi \Rightarrow \phi)} \big(\bigwedge_{i=1}^{n_1} (\overline{a_i} \vee A) \wedge \bigwedge_{j=1}^{n_2} (\overline{b_j} \vee A)\big) \tag{B.98}$$

and we see that no clauses in this derivation are wider than the sets of clauses in (B.96) and (B.97). Finally, we use $\big(\overline{a_i} \vee A\big)$ and $\big(\overline{b_j} \vee A\big)$ from (B.98) and $\big(\bigvee_{i=1}^{n_1} a_i \vee \bigvee_{j=1}^{n_2} b_j \vee A\big)$ from (B.96) to derive

$$\bigwedge_{A \in \mathit{CNF}(\Psi \Rightarrow \phi)} A = \mathit{CNF}\,(\Psi \Rightarrow \phi) \tag{B.99}$$

in $\big(n_1 + n_2\big) \cdot L\,(\mathit{CNF}\,(\Psi \Rightarrow \phi))$ steps and no extra width.

Summarizing, (B.99) is derivable from (B.96) and (B.97) in length

$$\big(2n_1 n_2 + n_1 + n_2\big) \cdot L\,(\mathit{CNF}\,(\Psi \Rightarrow \phi)) \leq 2k^{2|\Psi \Rightarrow \phi|+3} \tag{B.100}$$

and width

$$\max\left\{W\left(CNF\left(\Psi \Rightarrow \psi' \Rightarrow \phi\right)\right), W\left(CNF\left(\Psi \Rightarrow {\psi'}^{C} \Rightarrow \phi\right)\right)\right\} \leq \\ \leq \big(2|\Psi \Rightarrow \phi| + 2\big)\big(k-1\big). \quad \text{(B.101)}$$

The special cases that need to be taken care of are:

1. H_{i_1} and/or H_{i_2} are on the form $\Psi \Rightarrow \psi' \Rightarrow \bot$.

   The same proof as above holds (just replace clauses $A \in CNF\left(\Psi \Rightarrow \phi\right)$ with subclauses $B \in CNF\left(\Psi \Rightarrow \bot\right)$ where appropriate).

2. $\psi' = \bigvee_{i=1}^{n_1} a_i \equiv \top$ or $\psi' = \bigvee_{i=1}^{n_1} a_i \equiv \bot$.

   It is easy to adjust the proof of the general case given above (or of case 1) to a proof of the same bounds on length and width for this special case. Just simplify (B.96) and (B.97) and derive (B.99) without the intermediate step (B.98). We omit the details.

It follows that dilemma rule derivation steps can be derived in resolution within the bounds on length and width in theorem B.3. The proof of the theorem is completed.