A Framework Combining Diagrammatic and Symbolic Problem Solving

Konstantine Arkoudas and Selmer Bringsjord  
Rensselaer AI & Reasoning (RAIR) Lab  
Department of Cognitive Science  
Department of Computer Science  
Rensselaer Polytechnic Institute (RPI)  
Troy NY 12180 USA  
{arkouk,selmer}@rpi.edu  

November 28, 2007
Abstract

We introduce Vivid, a domain-independent framework for mechanized heterogeneous natural deduction that combines diagrammatic and symbolic reasoning. The framework is presented in the form of a family of denotational proof languages (DPLs). We present novel formal structures, called named system states, that are specifically designed for modeling potentially underdetermined diagrams. These structures allow us to deal with incomplete information, which is a pervasive feature of heterogeneous problem solving. We introduce a notion of attribute interpretations that enables us to interpret first-order signatures into named system states, and develop a formal semantic framework based on a three-valued logic. We extend the assumption-base semantics of DPLs to accommodate diagrammatic reasoning by introducing general inference mechanisms for the valid extraction of information from diagrams and for the incorporation of sentential information into diagrams. A rigorous big-step operational semantics is given, on the basis of which we prove that our framework is sound. We present examples of particular instances of Vivid, and discuss related work.

Keywords: Vivid, heterogeneous reasoning, diagrams, DPLs, assumption bases, named system states, worlds, three-valued logic.

1.1 Introduction

Diagrams have been recognized as valuable representational and reasoning tools at least since the days of Euclid. They are used extensively in a very wide range of fields. To note just a few examples, witness free-body, energy-level and Feynman diagrams in physics (Veltman 1995); arrow diagrams in algebra and category theory (Pierce 1991); Euler and Venn diagrams in elementary set theory; function graphs in calculus and analysis; planar figures in geometry; bar, chart, and pie graphs in economics; circuit and state timing diagrams in hardware design (Johnson, Barwise and Allwein 1996); UML diagrams in software design (Rumbaugh, Jacobson and Booch 1999); higraphs in specification (Harel 1988); visual programming languages (Chang 1990) and visual logic and specification languages (Agusti, Puigsegur and Robertson 1998, Hirakawa, Tanaka and Ichikawa 1990, Ogawa and Tanaka 2000); transition graphs in model checking (Bérard, Bidoit, Finkel, Laroussinie, Petit, Petrucci and Schnoebelen 2001); ER-diagrams and hypergraphs in databases (Fagin, Mendelzon and Ullman 1982); semantic networks in AI; icons and other pictorial devices in graphical user interfaces (GUIs) and information visualization (Mullet and Sano 1994, Tuft 1990, Ware 2004, Bier, Stone, Pier, Buxton and DeRose 1993); and so on. As the capability of computers to store and manipulate diagrams improves, their use is likely to increase.

The utility of diagrams is often thought to stem from the fact that diagrams have structural correspondences with the objects or situations they represent—they are analogical representations in the celebrated terminology of Sloman (1971), or homomorphic representations in the terminology of Barwise and Etchemendy (1995a); also see Sloman (1995) and Hayes (1985). To put it more plainly, a diagram resembles what the diagram depicts, in contrast to sentential—or “Fregean” (Sloman 1971)—descriptions. This was noticed at least as far back as the 19th century, when Peirce observed that a diagram is “naturally analogous to the thing represented” (1960, p. 316).

Consider, for instance, the task of describing some human face. We could perhaps describe the face with a collection of English sentences, or with a set of sentences in some formal language. But such a description is likely to be long and complicated, and not particularly illuminating. A drawing or a picture of the face, on the other hand, will be much more perspicuous, as well as significantly more compact than any sentential representation. Of course, some diagrams are better than others. A talented artist will produce a drawing that is a much more accurate depiction than the scrawlings of a child. A digital picture will be even more accurate. So, as Hammer (1995) observes, being an analogical or homomorphic representation is not a distinguishing feature of diagrams in general, but rather a distinguishing feature of good diagrams.

This ability of (good) diagrams is in turn often thought to derive from the fact that diagrams are two-dimensional objects, and therefore spatial relationships in the diagram can directly reflect analogous relationships in the underlying domain, an observation made a while back by Russell (1923). A classic example are maps. We can represent the streets of a city graphically, with a map, or sententially, e.g., by a collection of assertions expressing the various intersections and so forth. The graphical representation is without doubt a more intuitive and effective description because its spatial structure is similar to the actual layout of the city. This analogical correspondence is lost in the sentential representation.

The terms “sentential” and “symbolic” will be used synonymously throughout this paper.

Fractals (Manbelbrot 1982) might be able to yield compact representations for some complex shapes such as coastlines, etc., but the equations generating the fractals would be no more analogous to the corresponding shapes than other symbolic descriptions.

In the limiting case, the ultimate representation of an object is the object itself; in that case we have a perfect isomorphism between the representation and the object represented.
As another example, consider a map of a lake and try to imagine a sentential description of it. Stenning and Lemon (2001) trace this discrepancy to the fact that sentential languages derive from acoustic signals, which are one-dimensional and must therefore rely on a complex syntax for representation, something that is not necessary in the case of diagrams.

Nevertheless, two-dimensionality by itself is neither a necessary nor a sufficient condition for being a diagram. For instance, as Hammer (1995) points out, a representation of a picture by a two-dimensional array of numbers encoded under some encryption scheme does not count as a diagram; there is no structural similarity between the representation and that which is being represented. And by making sufficiently clever conventions, one can construct analogical one-dimensional diagrams. For example, the following string asserts that the stretch of road between Main Street/35th Street and Main/36th is two-way, whereas that between Main Street/36th and Main Street/37th is one-way and proceeds from right to left:

\[
\text{Main|35th} \iff \text{Main|36th} \iff \text{Main|37th}
\]

The foregoing points are worth stressing. In an era of Powerpoint and multimedia presentations, it is often taken for granted that graphical displays of information are automatically clearer and more intuitive than text, simply by virtue of being "visual." That is emphatically not the case. Diagrams are helpful only when their visual structure is analogical or homomorphic with the semantic structure of the information which they represent. The reason why Euler circles are intuitive, for instance, is precisely because spatial enclosure is naturally analogous to the subset relation, spatial overlap to set-theoretic intersection, and spatial separation to set-theoretic disjointness (Shin and Lemon 2003).

In the absence of such structural similarities, diagrams can quickly degenerate into what Tufte (1990, p. 34) calls "chartjunk": cluttered displays of lines, curves, arrows, bars, charts, and the like, that end up obscuring rather than clarifying information. Conversely, a diagram does not have to be visually arresting or elaborate in order to be superior to a sentential representation. It does not even have to be two-dimensional, as Hammer (1995, section 1.1) has noted, a point which is borne out by our Main Street example above, or by Hammer’s example of

the following one-dimensional diagram meant to express the relative distances between the Earth, Moon, and Mars when the Moon is aligned to fall between Earth and Mars:

\[
\text{Earth–Moon———Mars}
\]

This diagram is one dimensional: its syntax can be adequately modeled by sequences of symbols. (Hammer 1995, p. 2)

An early reviewer of this article criticized the diagrams in our paper as inordinately “simple” and “of a purely structural nature,” hence suffering from insufficient “diagrammaticity”—the implication being that only visually elaborate diagrams qualify as “truly” diagrammatic. The criticism is misguided, as it misses all of the above points. Indeed, consider the well-known example of the seating puzzle of Barwise and Etchemendy (1990), which we discuss extensively in Section 1.9. The diagrams in that puzzle are indeed extremely simple (one-dimensional, small, and purely ASCII); but they are no less powerful as a result. In fact, their structural nature and simplicity, far from being defects, are positively conducive to their representational power. Structure and simplicity are advantages of analogical representations, not disadvantages.

Even when diagrams are perspicuous analogical representations, however, their use is not entirely without drawbacks. While they often excel in depicting particular, concrete objects and situations, they are usually not as good for describing general, abstract structures and relationships. Spatial constraints tend to pull diagrams toward specificity, and end up limiting their generality and expressivity as a result. For instance, if we say that “two cities B and C are to the west of city A,” we make no commitment as to how B and C are positioned relative to each other, e.g., whether B is further west or east of C, whether both are on the same meridian but one is north of the other, etc. But any attempt to draw the proposition expressed by the foregoing statement would have to place B and C somewhere on the plane, and would therefore indicate a certain spatial relationship between them that was not present in the original sentence. That is what we mean when we say that spatial constraints tend to force diagrams to be specific, even when specificity is not intended.

Sentential descriptions are particularly superior—and indeed often necessary—when it comes to expressing complex propositions. It is easy enough to depict an atomic piece of information such as conveyed by the sentence “a is square” diagrammatically: We simply draw a square and label it a. But the proposition expressed by the statement “a is not square” is considerably more problematic. How do we draw something that is not square? Certainly drawing it as

\footnote{Peter Norvig provides an amusing but compelling illustration of this point in his Powerpoint version of the Gettysburg address, where he turns “four scores and seven years” into a gratuitous graph: http://www.norvig.com/Gettysburg/sld005.htm. More information can be found at http://www.norvig.com/Gettysburg/making.html.}
a triangle will not do, nor as a cylinder or as any other particular shape. We need to stipulate a specific graphical convention for signifying that an object is not square. Perhaps we could draw a square with a line over it, to indicate negation, but if there are other attributes in addition to shape, say color, then would a line over a red square negate only squarehood or redness as well? What if we only wanted to say that it is not red? Clearly, any conventions we make will be ad hoc solutions[^5] and can easily get out of hand. Conjunctions and disjunctions can also be problematic, particularly the former. Simply juxtaposing several different diagrams does not produce a single diagram as a result. And while clever abstraction tricks can be introduced to express disjunctive information diagrammatically, sometimes with great visual clarity[^6] for most domains it will be practically impossible to have enough abstraction tricks to be able to express arbitrary disjunctions. Existential and universal quantifications, being compactly expressed disjunctions and conjunctions, are even more powerful: “There is a knight on the chessboard” represents a huge number of possible chessboard configurations and excludes a huge number of others with just a few bits. In general, the issue is that in most interesting domains there are too many logical possibilities (models), while, due to physical spatial constraints, there is a much smaller number of possible diagrams. This discrepancy results in a certain tension. On one hand, the discrepancy works to the cognitive advantage of diagrams, since the fewer the graphical possibilities, the clearer the diagrams. (Indeed, if we keep adding conventions and abstraction tricks in order to achieve a bijection between the class of diagrams and the class of set-theoretic models, we will probably end up ruining whatever analogical benefit we might have had originally. That is the case for Peirce’s visual system for propositional logic, for instance, whose diagrams stand in a bijective relationship with logical sentences[^7]. On the other hand, the discrepancy can result in serious expressive limitations for diagrams.

Expressive limitations can sometimes lead to incorrect conclusions, since different models might be wrongly conflated (represented by the same diagram). This is a known issue, for example, with Euler circles (1768), as a consequence of Helly’s theorem in convex topology (Eggleston 1969). A simple illustration of the problem, due to Lemon and Pratt (1997), is the following: Consider four sets \( A, B, C, \) and \( D, \) any three of which have a non-empty intersection:

\[
A \cap B \cap C \neq \emptyset; \\
A \cap B \cap D \neq \emptyset; \\
B \cap C \cap D \neq \emptyset; \\
A \cap C \cap D \neq \emptyset.
\]

These are four perfectly consistent premises. But any Euler diagram that tried to depict these premises would lead to the incorrect conclusion that all four sets have a non-empty intersection (i.e. that \( A \cap B \cap C \cap D \neq \emptyset \)), which does not follow from the premises[^8]. The reason is that there is no way to draw four convex regions on the plane so that any three of the regions intersect without having all four of them intersect. Again, this is a consequence of Helly’s theorem. Similar negative results hold for other diagrammatic ways of depicting sets and relationships between them, such as Englebretsen’s (1992) linear diagrams; see Lemon (2002) for a thorough discussion.

The complexity of diagrammatic reasoning is another potential concern. Roughly, there are two types of diagrammatic inference. In one of them, exemplified by Euler circles, Venn diagrams, etc., inference is carried out by drawing appropriate diagrams. We then simply read off the appropriate bits of information from the constructed picture. This type of diagrammatic inference is summarized by the slogan “If you can draw it, it holds.”[^9] In the second type of diagrammatic reasoning, inference is carried out in a more traditional sense, by deriving new diagrams from other diagrams that are given as premises, perhaps in tandem with given symbolic information (as in Vivid), or by extracting symbolic information from given diagrams. Computational complexity issues have been investigated more extensively for the former type of diagrammatic reasoning. For example, it has been realized that results obtained in studying the complexity of topological inference (Grigni, Papadias and Papadimitriou 1995) have a direct bearing on the complexity of drawing diagrams such as Euler circles. It has been shown, e.g., that propositional reasoning with Euler sets is NP-hard, even though reasoning about the same domain can be done polynomially using other representations (Lemon 2002). In our own work, diagrammatic inference is based on reasoning with incomplete information and has a strong model-theoretic

[^5]: Stenning (2002) calls such conventions “abstraction tricks.”
[^6]: An example is the judicious use of questionmarks inside diagrams, an abstraction trick that is often used in Vivid diagrams.
[^7]: Shin and Lemon (2003) make a similar point about Peirce’s modification of Euler diagrams (the addition of X-sequences, etc.), writing that “the arbitrariness of its conventions and more confusing representations sacrificed the visual clarity which Euler’s original system enjoys.”
[^8]: As a counterexample, take \( A = \{1, 2, 4\}, B = \{2, 3, 4\}, C = \{1, 3, 4\}, \) and \( D = \{1, 2, 3\}. \)
[^9]: For instance, to check the validity of a syllogism with a Venn diagram, all we have to do is draw a figure that represents the premises of the syllogism. When finished, the picture itself will tell us whether or not the conclusion follows; nothing further needs to be done. Hence, inference in such cases stops with the representation of the premises. In customary reasoning, by contrast, inference only begins after the premises have been represented.
flavor, proceeding essentially by model elimination. This can often make proof checking considerably more computationally intensive than it is in the purely sentential case, where proofs can be checked very efficiently (in $O(n \log n)$ time in the case of $\mathcal{NDC}$ (Arkoudas 2001a), where $n$ is the size of the proof$^{10}$). It would appear, therefore, that visual inference, at least in some cases, can be significantly more expensive than corresponding sentential reasoning.

For these and other reasons, researchers have concluded that logical reasoning frameworks must be heterogeneous or hybrid (Barwise and Etchemendy 1995a, Myers 1994); they must support both diagrammatic and sentential modes of representation and reasoning, allowing users to freely combine the two. In the attempt to formulate a generic framework for heterogeneous reasoning, one naturally confronts the question of what type of diagrams to use. As Barwise and Etchemendy (1995a) correctly observe, it would be impossible to construct a domain-independent framework for diagrammatic reasoning that relied on a specific type of diagrams. What makes a class of diagrams appropriate—i.e., good analogical representations—for certain problems might make them inappropriate for others. For example, at different times electrical engineers use state diagrams, circuit diagrams, and timing diagrams to represent and reason about hardware as needed by the appropriate viewpoint at hand (control, logic gates, or timing, respectively). There is no single type of diagram that is uniformly appropriate.

Nevertheless, we observe that much of what we do when we reason with a combination of diagrammatic and symbolic information does not depend on how the diagrams are drawn or even on what they mean. In this paper we identify what is common in a great variety of instances of heterogeneous reasoning, and proceed to factor it out and extrapolate it into general formal linguistic constructs. In the resulting framework, the type of diagrams used may vary from application to application, but the principles by which we reason with diagrammatic and symbolic information remain the same. This is not unlike other separations that are familiar from traditional symbolic logic, where our vocabulary might vary from application to application (we have different constant, relation, and function symbols as dictated by the problem domain), and the interpretation of the atomic formulas that we can build from that vocabulary will also vary, but the general principles by which we reason with such formulas do not change. The resulting framework has a highly modular structure: A specific Vivid language is obtained by fixing a class of diagrams (this is done by providing a diagram parser), and an interpretation of the diagrams through an appropriate attribute structure. This is akin to the way in which the CLP($X$) scheme instantiates specific constraint logic programming languages by substituting different domains for $X$ (Jaffar and Lassez 1987).

In the next section we present the notation that we will use throughout this paper. Section 1.3 introduces the main conceptual tools that will be used to specify and investigate the semantics of Vivid: attribute structures, attribute systems, and attribute system states. Section 1.4 presents the notion of alternative state extensions, which is a key component of Vivid semantics, with an emphasis on issues of computational efficiency. In section 1.5 we develop the theoretical framework necessary for defining and analyzing Vivid. We introduce the notion of attribute interpretations, which enables us to evaluate first-order formulas with respect to attribute system states in accordance with a three-valued logic. We also introduce named system states, formalizing the idea that arbitrary names can appear inside diagrams. A number of useful results are listed here that are needed later for our soundness theorem.$^{12}$ Section 1.6 presents and discusses the abstract syntax and formal evaluation semantics of Vivid; our main soundness result is also stated here. In section 1.7 we present a Vivid solution to a well-known puzzle in heterogeneous reasoning, while section 1.10 illustrates the use of Vivid for reasoning about constraint problems (in this particular example, map coloring). Finally, in section 1.11 we discuss related work, and section 1.12 concludes.

### 1.2 Notation

For any sets $A$ and $B$, $A \setminus B$ denotes their set-theoretic difference:

$$A \setminus B = \{ x \in A \mid x \notin B \}.$$  

We write $(a; b)$ for the ordered pair that has $a$ and $b$ as its first and second component, respectively, $(a; b; c)$ for the triple of $a$, $b$, and $c$, etc. For any $n \geq 0$ objects $x_1, \ldots, x_n$, $[x_1 \cdots x_n]$ is the list that has $x_i$ as its $i^{th}$ element. Given a list $L = [x_1 \cdots x_n]$ and $i \in \{1, \ldots, n\}$, we write $L(i)$ to denote $x_i$. Further, for any such $L$ and object $x$, we define

$$\text{Pos}(x, L) = \{ i \in \{1, \ldots, n\} \mid x = x_i \}.$$  

---

$^{10}$NDC is a DPL on which the design of Vivid was based.

$^{11}$Of course in some contexts this can be viewed as an advantage, as time is often traded for space. Vivid deductions, for instance, are typically much more compact than corresponding sentential deductions.

$^{12}$We have omitted proofs for space reasons. They can be found in a longer version of the paper, at http://kryten.mm.rpi.edu/vivid.pdf.
1.3 Attribute structures and systems

Finally, for an arbitrary relation \( R \) for \( A \)

\[ \text{More precisely:} \]

\[ \text{Example 1:} \]

\[ \text{Another system based on the same attribute structure might consist of two clocks } c_1 \text{ and } c_2, \text{ perhaps indicating New York and Tokyo times, respectively:} \]

\[ \text{Thus, if } x \text{ does not occur in } L \text{ then } \text{Pos}(x, L) = \emptyset. \text{ If } A \text{ is a set, then } A^* \text{ is the set of all lists of elements of } A. \]

\[ \text{The empty list } [\] \text{ is a sublist of every list; no non-empty list is a sublist of } [\]; while a list of the form } L = [x_1 x_2 \cdots x_n] \text{ is a sublist of a list of the form } [y_1 y_2 \cdots y_m] \text{ iff } (1) \ x_1 = y_1 \text{ and } [x_2 \cdots x_n] \text{ is a sublist of } [y_2 \cdots y_m]; \text{ or } (2) \ x_1 \neq y_1 \text{ and } L \text{ is a sublist of } [y_2 \cdots y_m]. \text{ Strings (words) over an alphabet } \Sigma \text{ are simply lists of elements of } \Sigma. \text{ To adhere to customary string notation, we will usually denote the empty string by } \epsilon \text{ (instead of } []) \text{, and we will omit the enclosing brackets when writing non-empty strings, e.g. writing } a b c \text{ instead of } [a \ b \ c]. \]

\[ \text{For any set } A, \text{ we write } \mathcal{P}_{\text{fin}}(A) \text{ for the set of all finite subsets of } A. \text{ When } n \text{ is a positive integer, } A^n \text{ denotes the cartesian product} \]

\[ A_1 \times \cdots \times A_n, \]

\[ \text{i.e., the set of all lists of length } n \text{ with elements drawn from } A \]

\[ \text{Given a (partial) function } f : A \to B \text{ and elements } x \in A, y \in B, f[x \mapsto y] \text{ denotes that function from } A \to B \text{ which maps } x \text{ to } y \text{ and agrees with } f \text{ on every other } x' \in A. \text{ More precisely:} \]

\[ f[x \mapsto y] = \begin{cases} \{ (f \setminus \{(x; f(x))\}) \cup \{(x; y)\} \} & \text{if } f \text{ is defined for } x; \\ f \cup \{(x; y)\} & \text{otherwise.} \end{cases} \]

\[ \text{For } A' \subseteq A, f \upharpoonright A' \text{ denotes the restriction of } f \text{ on } A', \text{ i.e.,} \]

\[ f \upharpoonright A' = \{(x; y) \mid f(x) = y \text{ and } x \in A'\}. \]

\[ \text{Finally, for an arbitrary relation } R \subseteq A_1 \times \cdots \times A_n, D(R) \text{ denotes the set } \{A_1, \ldots, A_n\}. \]

1.3 Attribute structures and systems

**Definition 1:** An attribute structure is a pair \( \mathcal{A} = (\{A_1, \ldots, A_k\}; \mathcal{R}) \) consisting of a finite collection of sets \( A_1, \ldots, A_k \) called attributes; and a countable collection \( \mathcal{R} \) of computable relations, with \( D(R) \subseteq \{A_1, \ldots, A_k\} \) for each \( R \in \mathcal{R} \).

An attribute structure is thus a type of regular heterogeneous algebraic structure (Meinke and Tucker 1992, Wechler 1992), without any operators, whose carriers are called “attributes” for reasons that will become clear soon. We will assume that \( \mathcal{R} \) includes the identity relation on each attribute \( A_i \); \( \{(a; a) \mid a \in A_i\} \).

We also assume that there is a unique label \( l_i \) attached to each attribute \( A_i \) of a structure \( \mathcal{A} \). A label will serve as an alias for the corresponding attribute. Moreover, when the relations of \( \mathcal{A} \) are immaterial, we identify \( \mathcal{A} \) with its attributes. We can then write \( \mathcal{A} \) simply as \( l_1 : A_1, \ldots, l_k : A_k \), where \( l_i \) is the label of \( A_i \). We further assume that there is a canonical total ordering \( \prec_l \) on the labels (attributes): \( l_1 \prec_l l_2 \cdots \prec_l l_k \). The number of attributes \( k \) is the cardinality of \( \mathcal{A} \), denoted by \( |\mathcal{A}| \). An attribute can be infinite. If all attributes are finite, we say that \( \mathcal{A} \) has a finite basis.

**Definition 2:** Let \( \mathcal{A} \) be any attribute structure. An attribute system based on \( \mathcal{A} \), or \( \mathcal{A} \)-system, for short, is a pair \( \mathcal{S} = (\{s_1, \ldots, s_n\}; \mathcal{A}) \) consisting of a finite number \( n > 0 \) of objects \( s_1, \ldots, s_n \) and \( \mathcal{A} \). An attribute of \( \mathcal{A} \) may include some (perhaps all) of the objects \( s_1, \ldots, s_n \). If that is the case for at least one attribute, \( \mathcal{S} \) is called automorphic. We refer to the product \( n \cdot |\mathcal{A}| \) as the system’s power.

When \( \mathcal{A} \) is obvious from the context or immaterial, we drop references to it and speak simply of “systems” rather than “\( \mathcal{A} \)-systems.” Further, we assume there is a given binary relation \( \prec_s \) which totally orders the system objects: \( s_1 \prec_s s_2 \cdots \prec_s s_n \). When \( \prec_s \) is inconsequential, we will not bother to specify it.

**Example 1:** Consider a system consisting of a clock \( c \), with two attributes, hours and minutes:

\[ \{c\}; \text{hours : }\{0, \ldots, 23\}, \text{minutes : }\{0, \ldots, 59\}. \]

Another system based on the same attribute structure might consist of two clocks \( c_1 \) and \( c_2 \), perhaps indicating New York and Tokyo times, respectively:

\[ \{c_1, c_2\}; \text{hours : }\{0, \ldots, 23\}, \text{minutes : }\{0, \ldots, 59\}. \]

\[ ^{13}\text{With } A^1 = A. \]
Example 2: Consider a system comprising the nodes of a three-element linked list, each with two attributes, a data field consisting of a Boolean value (t or f) and a next field consisting of another node or the null value:

\[
\{\{n_1, n_2, n_3\}; \text{data : Bool, next : } \{n_1, n_2, n_3, \text{null}\}\},
\]

where \(\text{Bool} = \{\text{t, f}\}\) and null is a special token distinct from \(\{n_1, n_2, n_3\}\). This is an automorphic system. ■

Example 3: Consider a blocks-world system consisting of three blocks \(A, B,\) and \(C\), and a single “position” attribute, where a position is either a block or the floor:

\[
\{\{A, B, C\}; \text{pos : } \{A, B, C, \text{floor}\}\},
\]

and floor is distinct from \(A, B,\) and \(C\). This system is also automorphic. ■

Example 4: Consider a Hyperproof (Barwise and Etchemendy 1995b) system consisting of four blocks and three attributes: a pair of integers \((i; j)\) with \(0 < i, j < 9\) indicating a grid location; a size (small, medium, or large); and a shape (cube, tetrahedron, or dodecahedron):

\[
\{\{b_1, b_2, b_3, b_4\}; \text{loc : } \{1, \ldots, 8\}^2, \text{size : } \{\text{small, medium, large}\}, \text{shape : } \{\text{cube, tet, dodec}\}\}.
\]

Definition 3: A state of a system \(S = (\{s_1, \ldots, s_n\}, \{A_1, \ldots, A_k\})\) is a set of functions \(\sigma = \{\delta_1, \ldots, \delta_k\}\), where each \(\delta_i\) is a function from \(\{s_1, \ldots, s_n\}\) to the set of all non-empty finite subsets of \(A_i\), i.e.,

\[
\delta_i : \{s_1, \ldots, s_n\} \rightarrow \mathcal{P}_{\text{fin}}(A_i) \setminus \emptyset.
\]

We refer to each \(\delta_i\) as the state’s ascription into \(A_i\). An ascription \(\delta_i\) is a valuation if it maps every object to a singleton, i.e., if \(|\delta_i(s_j)| = 1\) for every \(j = 1, \ldots, n\). We may thus view a valuation as mapping every object to a unique attribute value. A world \(w\) is a state in which every ascription is a valuation. ■

A system for an attribute structure with a finite basis has

\[
\prod_{i=1}^{k} \left[2^{\lvert A_i \rvert} - 1\right]^n
\]

states, where \(n\) is the number of objects and \(k\) the number of attributes. To simplify notation, when \(\delta\) is a valuation that maps an object \(s\) to a singleton \(\{a\}\), we might write \(\delta(s) = a\) instead of \(\delta(s) = \{a\}\). Thus in some cases we will treat \(\delta(s)\) as an attribute value rather than a singleton comprising such a value; the context will always make this clear. Further, we will often use the label \(l_i\) of an attribute \(A_i\) to denote the corresponding ascription into \(A_i\). That is, we are overloading the label symbols: sometimes \(l_i\) will stand for the attribute \(A_i\) and sometimes, in the context of a given state, it will stand for \(\delta_i\), the state’s (unique) ascription into \(A_i\); again, the context will always make our intentions obvious. As an additional convention, given a state \(\sigma\) of the form described in Definition 3, an attribute (label) \(l_i\) and an object \(s_j\), we write \(\sigma(l_i, s_j)\) for \(\delta_i(s_j)\), i.e., the value of the ascription \(\delta_i\) for the object \(s_j\).

Example 5: Consider the single-clock system of Example 1:

\[
\{\{c\}; \text{hours : } \{0, \ldots, 23\}, \text{minutes : } \{0, \ldots, 59\}\}.
\]

A state \(\sigma_1\) of this system is given by the following two valuations:

\[
\sigma_1 : \text{hours}(c) = 15, \text{minutes}(c) = 47,
\]
indicating a time of 3:47 p.m. This is a particular world of the clock system. Using the aforementioned convention, we can also write:
\[ \sigma_1(\text{hours}, c) = 15, \sigma_1(\text{minutes}, c) = 47. \]

Suppose we know that it is between 2:30 a.m. and 3 a.m., but do not know exactly how many minutes past 2:30 it is. This knowledge can be captured by the following state:
\[ \sigma_2 : \text{hours}(c) = 2, \text{minutes}(c) = \{31, \ldots, 59\}. \]

This state can also be expressed by writing
\[ \sigma_2(\text{hours}, c) = 2, \sigma_2(\text{minutes}, c) = \{31, \ldots, 59\}. \]

Complete lack of information about the time is represented by the state:
\[ \text{hours}(c) = \{0, \ldots, 23\}, \text{minutes}(c) = \{0, \ldots, 59\}. \]

Example 6: Consider the linked-list system of Example 2. The state
\[ \text{data}(n_1) = t, \text{data}(n_2) = f, \text{data}(n_3) = t, \text{next}(n_1) = n_2, \text{next}(n_2) = n_3, \text{next}(n_3) = \text{null} \]
depicts the world shown in Figure 1.1. The state
\[ \text{data}(n_1) = \{t, f\}, \text{data}(n_2) = \{t, f\}, \text{data}(n_3) = f, \text{next}(n_1) = n_2, \text{next}(n_2) = \{n_1, n_3\}, \text{next}(n_3) = \text{null} \]
depicts a system in which we do not know the data fields of the first and second nodes, we know that the next field of the second node is either \( n_1 \) or \( n_3 \), and we have fixed values for the remaining nodes and attributes.

Example 7: Consider the blocks world system of Example 3. The state
\[ \text{pos}(A) = B, \text{pos}(B) = \text{floor}, \text{pos}(C) = \text{floor} \quad (1.1) \]
depicts the blocks world shown in Figure 1.2. The state
\[ \text{pos}(A) = \{A, B, C, \text{floor}\}, \text{pos}(B) = \{A, B, C, \text{floor}\}, \text{pos}(C) = \{A, B, C, \text{floor}\} \]
signifies complete lack of information about the positions of the blocks.

Here we assume that we are only interested in how the blocks are stacked, not in the left-of relation. So, in the case of Figure 1.2 if \( C \) were on the floor to the left of \( A \) and \( B \) instead of the right, we would still get the same state, (1.1). Of course we could enrich our attribute structure to take such additional information into account, if doing so was deemed important. In general, most abstract representations of a diagram will discard certain bits of information conveyed by the diagram as irrelevant. Which aspects of a diagram are considered essential and which are accidental depends on the domain at hand and our purposes. The issue is not with the representations but with the diagrams themselves. As discussed in the introduction, diagrams tend to be overly specific and thus we usually need to ignore at least some of their aspects.\(^1\)

Example 8: Consider the Hyperproof system of Example 4. The state
\[
\begin{align*}
\text{loc}(b_1) &= \{(1; 1)\}, & \text{size}(b_1) &= \{\text{small, medium}\}, & \text{shape}(b_1) &= \text{cube} \\
\text{loc}(b_2) &= \{(5; 3)\}, & \text{size}(b_2) &= \text{small}, & \text{shape}(b_2) &= \text{tet} \\
\text{loc}(b_3) &= \{(2; 6)\}, & \text{size}(b_3) &= \text{large}, & \text{shape}(b_3) &= \{\text{tet, dodec}\} \\
\text{loc}(b_4) &= \{(7; 1), (7; 2), \ldots, (7; 8)\}, & \text{size}(b_4) &= \text{medium}, & \text{shape}(b_4) &= \text{dodec}
\end{align*}
\]
should be self-explanatory at this point.

\(^1\)For instance, in the usual diagrammatic proof of the Pythagorean theorem, we must ignore the relative lengths of the triangle’s sides in order to ensure that we are reasoning about an “arbitrary” triangle, even though any picture of a triangle will necessarily depict specific lengths for its sides.
We might think of system states as mental models of situations (Johnson-Laird 1983), representing various states of knowledge ranging from completely specific to completely indeterminate.

Using the canonical orders on attributes and system objects, \( l_1 \prec_l \cdots \prec_l l_k \) and \( s_1 \prec_s \cdots \prec_s s_n \), a world \( w \) can be regarded as the unique finite string of the following form:

\[
w(l_1, s_1) \circ w(l_2, s_1) \cdots \circ w(l_k, s_1) \cdots w(l_1, s_n) \cdots w(l_k, s_n).
\]

A string of this form describes, for each attribute, the unique attribute value that every system object has in \( w \), in accordance with the canonical orderings on objects and attributes: First we have the attribute values of the first object, then the attribute values of the second object, and so on. Thus the length of any such string is \( k \cdot n \), i.e., equal to the power of the system.

**Example 9:** Assuming that data \( \prec_l \) next and that \( n_1 \prec_s n_2 \prec_s n_3 \), the world described in Example 6 (and shown in Figure 1.1) can be viewed as the six-element “string” \( t n_2 f n_3 t null \).

**Definition 4:** Consider a system \( S = (\{ s_1, \ldots, s_n \}; l_1 : A_1, \ldots, l_k : A_k) \). We say that a state \( \sigma' \) of \( S \) is an extension of another such state \( \sigma \), written \( \sigma' \sqsubseteq \sigma \), iff \( \sigma'(l_i, s_j) \subseteq \sigma(l_i, s_j) \) for every \( i = 1, \ldots, k \) and \( j = 1, \ldots, n \). We call \( \sigma' \) a proper extension of \( \sigma \), denoted \( \sigma' \sqsubset \sigma \), iff \( \sigma' \sqsubseteq \sigma \) and \( \sigma \not\sqsubseteq \sigma' \).

Hence, \( \sigma' \) is a proper extension of \( \sigma \) iff \( \sigma' \sqsubseteq \sigma \) and there is at least one attribute \( l \) and object \( s \) such that \( \sigma'(l, s) \subset \sigma(l, s) \). Worlds do not have any proper extensions.

Consider, for instance, the system of Example 1:

\[
\{ \{ c_1, c_2 \}; \text{hours} : \{0, \ldots, 23\}, \text{minutes} : \{0, \ldots, 59\} \}.
\]

The state

\[
\begin{align*}
\text{hours}'(c_1) &= \{13, 14\}, \\
\text{minutes}'(c_1) &= \{55\}, \\
\text{hours}'(c_2) &= \{6, 7\}, \\
\text{minutes}'(c_2) &= \{9, 10\},
\end{align*}
\]

is an extension (a proper one) of the state

\[
\begin{align*}
\text{hours}(c_1) &= \{13, 14, 15\}, \\
\text{minutes}(c_1) &= \{55\}, \\
\text{hours}(c_2) &= \{6, 7\}, \\
\text{minutes}(c_2) &= \{9, 10, 11\}.
\end{align*}
\]

Recalling that a world can be encoded as a string, we may view a system state \( \sigma \) as a language, namely as the set of all strings \( w \) such that \( w \sqsubseteq \sigma \). For instance, the second state described in Example 6 can be understood as the following set of eight strings:

\[
\begin{align*}
\{ t n_2 t n_1 f null, t n_2 t n_3 f null, t n_2 f n_1 f null, t n_2 f n_3 f null, \\
f n_2 t n_1 f null, f n_2 t n_3 f null, f n_2 f n_1 f null, f n_2 f n_3 f null \}.
\end{align*}
\]
Note that all such languages are finite, and therefore regular. This will be important in representing system states as minimal acyclic deterministic finite automata, which support highly efficient insertions and membership tests.

**Definition 5:** Two system states \( \sigma_1 \) and \( \sigma_2 \) are disjoint iff there is no world \( w \) such that \( w \subseteq \sigma_1 \) and \( w \subseteq \sigma_2 \).

Viewing \( \sigma_1 \) and \( \sigma_2 \) as languages, the two are disjoint iff they are set-theoretically disjoint, i.e., \( \sigma_1 \cap \sigma_2 = \emptyset \). Any two distinct worlds are automatically disjoint.\(^{16}\) Both perspectives on system states (as indexed sets of ascriptions and as languages) are useful, and both will be used throughout, as appropriate. For instance, we might speak of a world \( w \) as extending a certain state \( \sigma \), \( w \subseteq \sigma \), viewing \( w \) and \( \sigma \) as sets of ascriptions; or we might speak of \( w \) as belonging to \( \sigma \), \( w \in \sigma \), viewing the latter as a language and \( w \) as a string in that language.

The set of all states of \( S \) is arranged into a rich partial order corresponding to the join (union) semi-lattice

\[
\left( \mathcal{P}_{\text{fin}}(A_1) \setminus \emptyset \right) \times \cdots \times \left( \mathcal{P}_{\text{fin}}(A_k) \setminus \emptyset \right).
\]

We do not quite have a lattice, because the meet of two states might not exist. That is directly related to the proviso of Definition\(^{3}\) that ascriptions must map system objects to non-empty sets of attribute values, and ultimately stems from the expressive limitations of pictures. Given that incomplete information is part and parcel of our system, a join operator \( \sqcup \) on diagrams is fairly natural: For any attribute \( l \) and object \( s \), we set

\[
(\sigma_1 \sqcup \sigma_2)(l, s) = \sigma_1(l, s) \cup \sigma_2(l, s).
\]

This is precisely the least upper bound of the two states w.r.t. the ordering \( \sqsubseteq \). But a meet operator \( \sqcap \) would indicate conjunction, and conjoining diagrams with contradictory information is not pictorially meaningful. For instance, if an object \( s \) has a round shape in diagram \( \sigma_1 \) and a square shape in \( \sigma_2 \):

\[
\begin{align*}
\sigma_1(\text{shape}, s) &= \text{round}; \\
\sigma_2(\text{shape}, s) &= \text{square};
\end{align*}
\]

then what is the shape of \( s \) in \( \sigma_1 \cap \sigma_2 \)? If we define meets as

\[
(\sigma_1 \cap \sigma_2)(l, s) = \sigma_1(l, s) \cap \sigma_2(l, s),
\]

then we would have \( \sigma_1 \cap \sigma_2(\text{shape}, s) = \emptyset \), an impossible state of affairs. Sententially, we can easily write down a formula that asserts

\[
\text{Round}(s) \land \text{Square}(s),
\]

but, diagrammatically, we cannot draw a square circle. Likewise, we can very well write a sentence that asserts that it is currently 5:15 a.m. and also that it is currently 11:30 p.m., but if I look at the face of a clock I will not see it displaying both times. There are no inconsistent diagrams\(^{17}\) and this is, in turn, due to the fact that negation is not diagrammatically meaningful. If we had a negation operator \( \neg \) on diagrams then conjunction could be defined simply as \( \sigma_1 \cap \sigma_2 = -(\sigma_1 \sqcup \sigma_2) \). But negating a diagram could of course take us to the empty set if the starting value comprised the entire attribute space.

### 1.4 Alternative state extensions

In this section we introduce the notion of alternative system state extensions and prove some results which play an important role in the formal semantics of Vivid.

**Definition 6:** Let \( \sigma_1, \sigma_2 \) be proper extensions of a state \( \sigma \). We say that \( \sigma_2 \) is an alternative extension of \( \sigma \) with respect to \( \sigma_1 \), written \( \text{Alt}(\sigma, \sigma_1, \sigma_2) \), iff there is an attribute \( l \) and an object \( s \) such that:

1. \( \sigma_1(l, s) \subset \sigma(l, s) \);
2. \( \sigma_2(l, s) = \sigma(l, s) \setminus \sigma_1(l, s) \); and
3. for all attributes \( l' \) and objects \( s' \), if \( l' \neq l \) or \( s' \neq s \) then \( \sigma_2(l', s') = \sigma(l', s') \).

\(^{16}\)When a world is viewed as a language, it is obviously understood as a singleton.

\(^{17}\)Optical illusions should not be confused with logical inconsistency.
It follows immediately that if such an attribute and object exist, they must be unique.

As a simple example, consider a system consisting of one object $s$ with two attributes, color and size, and suppose that $\sigma$ stipulates red, green, and blue as the possible color values of $s$ and large, medium and small as its possible size values; and suppose that $\sigma_1$ extends $\sigma$ by limiting the color values of $s$ to green and blue and its size to small:

What counts as an alternative extension of $\sigma$ w.r.t. $\sigma_1$? Considering that $\sigma_1$ essentially states that the color of $s$ is either green or blue and that its size is small, we could differ from it in one of the following respects:

<table>
<thead>
<tr>
<th>Color of $s$</th>
<th>Size of $s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>{red}</td>
<td>{large, medium}</td>
</tr>
<tr>
<td>red</td>
<td>small</td>
</tr>
<tr>
<td>green, blue</td>
<td>{large, medium}</td>
</tr>
<tr>
<td>red</td>
<td>{large, medium}</td>
</tr>
</tbody>
</table>

That is, we could either choose to (1) disagree with the color, and either disagree or agree with the size (the latter choice is immaterial in light of the first disagreement), resulting in the top two rows of the table above, or (2) disagree with the size and either agree or disagree with the color (again, this being immaterial), which leads to the third and fourth rows. Given that set membership represents disjunctive information, we can collapse the first two and last two possibilities, obtaining the following two states, $\sigma_2$ and $\sigma_3$:

$\sigma_2$: | Color of $s$ | Size of $s$ |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{red}</td>
<td>{small, large, medium}</td>
</tr>
</tbody>
</table>

$\sigma_3$: {red, green, blue} | {large, medium} |

These are the only two alternative extensions of $\sigma$ w.r.t. $\sigma_1$. In general, given an arbitrary extension $\sigma_1 \sqsubseteq \sigma$, we can effectively construct all alternative extensions of $\sigma$ w.r.t. $\sigma_1$. There are $m$ such extensions, where $m$ is the number of attribute-object pairs (or a.o. pairs for short) $(l; s)$ such that $\sigma_1(l, s) \neq \sigma(l, s)$, or equivalently, such that

$\sigma_1(l, s) \subset \sigma(l, s)$;

i.e., the number of pairs of attributes and objects whose corresponding ascription values changed in going from $\sigma$ to $\sigma_1$.

We can generate the alternative states by taking the complement of the ascription value of each such pair in $\sigma$ (clause 2 of Definition 3) while reverting the other $m - 1$ pairs to their $\sigma$ values (clause 3 of Definition 3). We stress that in determining the alternative extensions of $\sigma$ w.r.t. $\sigma_1$ we only consider those objects and attributes that are changed by $\sigma_1$. We ignore those ascription assignments that remain the same in going from $\sigma$ to $\sigma_1$. As another example, there are two states that are alternative extensions of (1.3) w.r.t. (1.2). In one of them we keep the original hours values of $c_1$ ($\{13, 14, 15\}$) but complement the minutes value of $c_2$ to obtain $\{11\}$; while in the other alternative we keep the original minutes value of $c_2$ ($\{9, 10, 11\}$) but complement the hours value of $c_1$ to obtain $\{15\}$. In both cases the minutes of $c_1$ and hours of $c_2$ remain the same as in the original state (1.3), as neither of them was modified by (1.2).

As we remarked earlier, a system state can be viewed as a finite language $L$, where each string in $L$ represents a world. This means that states can be represented by tries, using the canonical orderings on attributes and system objects. For instance, suppose that in the running example we have color < size, and that we use $R$, $G$, and $B$ as abbreviations for red, green, and blue, and $S$, $M$, and $L$ for the three sizes. Then the states $\sigma$ and $\sigma_1$ can be represented by the tries shown.
in Figure 1.3. This makes the language-theoretic interpretation of system states easier to visualize, and allows for very efficient computation of optimal sets of alternative extensions, as we will explain shortly.

In particular, the worlds that extend a state $\sigma$ are precisely the maximal paths in the trie representation of $\sigma$. That is, $w \sqsupseteq \sigma$ iff $w$ is a branch in the trie representing $\sigma$. Alternatively, the strings encoded by the trie are the elements of the language $\sigma$, which is to say all and only the worlds which extend $\sigma$. In the case of the state $\sigma$ of our current example (the left trie in Figure 1.3), we have:

$$\sigma = \{RS, RM, RL, GS, GM, GL, BS, BM, BL\}.$$ 

These strings encode all and only the worlds which are captured by $\sigma$. The first element of each string represents the color of object $s$ and the second its size. Since there are three possibilities for color and three for size, there are nine possible worlds altogether, which are exactly the strings that are encoded by the nine branches of the try, where the first level of the trie represents the choice of color and the second the choice of size. Note that the trie for $\sigma_1$, on the right side of Figure 1.3, is a subtree of the trie on the left, reflecting the fact that $\sigma_1 \sqsubseteq \sigma$, i.e., viewing $\sigma$ and $\sigma_1$ as languages, $\sigma_1 \subset \sigma$.

Now consider $\sigma_2$ and $\sigma_3$ again, the two alternative extensions of $\sigma$ w.r.t. $\sigma_1$, as computed by the algorithm we described above:

$$\begin{align*}
\sigma_2(\text{color}, s) &= \{R\} \\
\sigma_2(\text{size}, s) &= \{S, M, L\} \\
\sigma_3(\text{color}, s) &= \{R, G, B\} \\
\sigma_3(\text{size}, s) &= \{M, L\}
\end{align*}$$

These two states are suboptimal in the sense that they do not convey unique information—there are worlds which are represented by both of them. In the terminology of Definition 5, the states are not disjoint. This becomes clear if we depict them as tries, which is done in Figure 1.4. As that figure makes clear, the strings (worlds) $RM$ and $RL$ in $\sigma_3$ are also elements of $\sigma_2$. These strings represent the possibilities that (a) $s$ has red color and is of medium size; and (b) $s$ has...
red color and is large. These two possibilities are represented both by $\sigma_2$ and by $\sigma_3$. This can also be seen by noticing that the subtree in the box with the dashed lines is a subtree of the $\sigma_2$ trie on the left. In the interest of efficiency, such redundancies should be eliminated so that all computed alternative extensions are pairwise disjoint. This is important not only in terms of minimizing the storage allocated for the alternative extensions but also because, as we will see later, the semantics of Vivid frequently require the repeated evaluation of certain formulas in a set of alternative extensions of a given state, so, to ensure that no redundant work is performed, we need to store only disjoint alternative extensions. This could be achieved incrementally by maintaining a trie $T$ representing the union of all computed alternative extensions. As a new alternative extension $\sigma'$ is being computed (via the preceding algorithm), we check to see if each world $w$ in $\sigma'$ is also in $T$. If $w$ is already in $T$, it is not added to $\sigma'$; otherwise it is added both to $\sigma'$ and to $T$. This method is quite efficient, since string insertions and lookups in tries are extremely fast.

Nevertheless, a more compact representation of the tries can be obtained via deterministic finite automata (DFA). Since every language that corresponds to a system state is finite, the DFAs in question are always acyclic, i.e., they are, from a graph-theoretic perspective, directed acyclic graphs (DAGs). The acyclic deterministic finite automaton (ADFA) recognizing $\sigma$, for instance, is shown in Figure 1.5. As a graph, it is essentially the DAG version of the trie on the left of Figure 1.3. The ADFA for $\sigma_1$ is:

![ADFA for $\sigma_1$](image)

while the ADFAs for the two alternative extensions $\sigma_2$ and $\sigma_3$ are, respectively:

![ADFA for $\sigma_2$](image)

and

![ADFA for $\sigma_3$](image)

There are very efficient incremental algorithms for constructing minimal ADFAs, even when the incoming strings are not sorted (Daciuk, Mihov, Watson and Watson 2000). There are efficient algorithms for converting a trie into a minimal ADFA as well, but clearly tries can have excessive memory requirements. In our case there is no need to build a try in
the first place, so minimal ADFAs representing alternative extensions can be constructed on the fly. As a new alternative extension \( \sigma' \) is computed (again, via the algorithm we described earlier), we check to see whether each world \( w \in \sigma' \) is accepted by some previous ADFA (encoding one of the alternative extensions that have already been computed). If so, the world is not inserted into the ADFA for \( \sigma' \); otherwise it is. In this example, for instance, the strings \( RM \) and \( RL \) would not be added to the ADFA for \( \sigma_3 \) because they are already accepted by the ADFA encoding \( \sigma_2 \).

As can be seen pictorially either from the tries of Figure 1.3 and Figure 1.4, or from the ADFAs shown above, the union of the strings in \( \sigma_2 \) and \( \sigma_3 \) is precisely the complement of \( \sigma \) w.r.t. \( \sigma_1 \):

\[
\sigma = \{ RS, RM, RL, GS, GM, GL, BS, BM, BL \};
\sigma_1 = \{ GS, BS \};
\sigma_2 = \{ RS, RM, RL \};
\sigma_3 = \{ GM, GL, BM, BL \}.
\]

The following result proves that this is indeed always the case, and is the main idea behind the notion of alternative extensions:

**Lemma 1:** If \( w, \sigma' \subseteq \sigma \) and \( w \not\subseteq \sigma' \) then there is some \( \sigma'' \subseteq \sigma \) such that \( \text{Alt}(\sigma, \sigma', \sigma'') \) and \( w \subseteq \sigma'' \). In words: if a world \( w \) and a state \( \sigma' \) both extend \( \sigma \) and \( w \) does not extend \( \sigma' \), then there is an alternative extension \( \sigma'' \) of \( \sigma \) w.r.t. \( \sigma' \) such that \( w \) extends \( \sigma'' \).

**Proof:** Since both \( \sigma' \) and \( w \) are extensions of \( \sigma \), we have

\[
\sigma'(l, s) \subseteq \sigma(l, s) \tag{1.4}
\]

and

\[
w(l, s) \subseteq \sigma(l, s) \tag{1.5}
\]

for every attribute \( l \) and system object \( s \). Further, since \( w \not\subseteq \sigma' \), there exist an attribute \( l_i \) and an object \( s_j \) such that \( w(l_i, s_j) \not\subseteq \sigma'(l_i, s_j) \), i.e., there is some attribute value \( \alpha \) such that

\[
\alpha \in w(l_i, s_j) \tag{1.6}
\]

and

\[
\alpha \notin \sigma'(l_i, s_j). \tag{1.7}
\]

Moreover, since \( w \) is a world we have \( w(l_i, s_j) = \{ \alpha \} \). From (1.6) and (1.5) we infer

\[
\alpha \in \sigma(l_i, s_j). \tag{1.8}
\]

From (1.8), (1.7), and (1.4) we obtain

\[
\sigma'(l_i, s_j) \subset \sigma(l_i, s_j). \tag{1.9}
\]

Now define \( \sigma'' \subset \sigma \) as follows:

\[
\sigma''(l_i, s_j) = \sigma(l_i, s_j) \setminus \sigma'(l_i, s_j) \tag{1.10}
\]

while for every attribute \( l \) and object \( s \) such that \( l \neq l_i \) or \( s \neq s_j \), set

\[
\sigma''(l, s) = \sigma(l, s) \tag{1.11}
\]

It follows by construction (specifically, from (1.9), (1.10), and (1.11)) that \( \text{Alt}(\sigma, \sigma', \sigma'') \). Further, \( w \subseteq \sigma'' \). To see this, consider any attribute \( l \) and object \( s \). Either \( l = l_i \) and \( s = s_j \), or not. In the former case we have \( w(l, s) = \{ \alpha \} \), so from (1.10), (1.8), and (1.7) we conclude \( \alpha \in \sigma''(l, s) \), hence \( w(l, s) \subseteq \sigma''(l, s) \). In the latter case, \( w(l, s) \subseteq \sigma''(l, s) \) follows from (1.11) and (1.5).

We now generalize the foregoing notion of alternative extensions so that it obtains w.r.t. to several states instead of just one. We will see that the new definition (Definition below) subsumes the one given above.

\[\text{Adapted: } w(l, s) \subseteq \sigma''(l, s) \]

Alternatively, we could incrementally construct one single minimal ADFA representing all alternative extensions. This might be preferable when the number of alternative extensions is very large.
Definition 7: A list of \(m \geq 1\) a.o. pairs \([(l_1; s_1) \cdots (l_m; s_m)]\) is **homogeneous** iff \(l_1 = \cdots = l_m\) and \(s_1 = \cdots = s_m\), i.e., iff all \(m\) pairs are identical.

Definition 8: Let \(\sigma_1, \ldots, \sigma_m \sqsubset \sigma, m \geq 1\). A list of \(m\) a.o. pairs \(L = [(l_1; s_1) \cdots (l_m; s_m)]\) **spans** the states \(\sigma_1, \ldots, \sigma_m\) with respect to \(\sigma\) iff

\[
\sigma_i(l_i, s_i) \subset \sigma(l_i, s_i)
\]

for every \(i = 1, \ldots, m\). In addition, we say that \(L\) **properly spans** \(\sigma_1, \ldots, \sigma_m\) w.r.t. \(\sigma\) iff for every sublist \([i_1 \cdots i_m']\) of \([1 \cdots m]\) such that \([L(i_1) \cdots L(i_m')]\) is homogeneous, we have

\[
\bigcup_{j=1}^{m'} \sigma_{i_j}(l_{i_j}, s_{i_j}) \subset \sigma(l_{i_j}, s_{i_j}).
\]

Equivalently, \(L\) does not properly span \(\sigma_1, \ldots, \sigma_m\) with respect to \(\sigma\) iff for some such sublist we have

\[
\sigma_i(l_i, s_i) \cup \cdots \cup \sigma_{i_m'}(l_{i_m'}, s_{i_m'}) = \sigma(l_i, s_i).
\]

Note that every list of length one that spans \(\sigma_1\) w.r.t. \(\sigma\) (for \(\sigma_1 \sqsubset \sigma\)) does so properly. That is why the definition below is a proper generalization of Definition 8.

Definition 9: Let \(\sigma_1, \ldots, \sigma_m, \sigma' \sqsubset \sigma, m \geq 1\). We say that \(\sigma'\) is an **alternative extension of \(\sigma\) w.r.t. \(\sigma_1, \ldots, \sigma_m\)**, written \(Alt(\sigma, \{\sigma_1, \ldots, \sigma_m\}, \sigma')\), iff there is a list \(L = [(l_1; s_1) \cdots (l_m; s_m)]\) properly spanning \(\sigma_1, \ldots, \sigma_m\) w.r.t. \(\sigma\) such that for every attribute \(l\) and object \(s\) we have

\[
\sigma'(l, s) = \sigma(l, s) \setminus \bigcup_{i \in Pos((l; s), L)} \sigma_i(l, s).
\]

We write \(\text{AE}(\{\sigma_1, \ldots, \sigma_m\}, \sigma)\) for the set of all alternative extensions of \(\sigma\) w.r.t. \(\sigma_1, \ldots, \sigma_m\).

Therefore, to compute all alternative extensions of \(\sigma\) w.r.t. \(\sigma_1, \ldots, \sigma_m\) we need to compute all lists of a.o. pairs that properly span \(\sigma_1, \ldots, \sigma_m\) w.r.t. \(\sigma\). We will present algorithms for both tasks shortly, but we first turn to an example that will help to clarify these definitions.

Example 10: Suppose we have two objects \(s_1\) and \(s_2\), with \(s_1 \prec_s s_2\), and two attributes, \(\text{color} \prec \text{size}\), with color having three possible values (red, green and blue, abbreviated R, G, and B), and with size also having three possible values (S, M, and L). Suppose further that the starting state \(\sigma\) is as follows:

<table>
<thead>
<tr>
<th>(\sigma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>color((s_1) = {R, B})</td>
</tr>
<tr>
<td>size((s_1) = {S, M, L})</td>
</tr>
<tr>
<td>color((s_2) = {R, B, G})</td>
</tr>
<tr>
<td>size((s_2) = {M, L})</td>
</tr>
</tbody>
</table>

Now consider the following three proper extensions of \(\sigma\):

<table>
<thead>
<tr>
<th>(\sigma_1)</th>
<th>(\sigma_2)</th>
<th>(\sigma_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>color((s_1) = {B})</td>
<td>color((s_1) = {R, B})</td>
<td>color((s_1) = {R})</td>
</tr>
<tr>
<td>size((s_1) = {S, M})</td>
<td>size((s_1) = {L})</td>
<td>size((s_1) = {S, M, L})</td>
</tr>
<tr>
<td>color((s_2) = {B, G})</td>
<td>color((s_2) = {R, B, G})</td>
<td>color((s_2) = {R, B, G})</td>
</tr>
<tr>
<td>size((s_2) = {M, L})</td>
<td>size((s_2) = {L})</td>
<td>size((s_2) = {M, L})</td>
</tr>
</tbody>
</table>

We have used the labels A—F to mark those a.o. pairs \((l; s)\) for which state \(\sigma_i\) properly extends \(\sigma\), i.e., such that \(\sigma_i(l, s) \subset \sigma(l, s)\). The following lists of a.o. pairs span \(\sigma_1\), \(\sigma_2\), and \(\sigma_3\) w.r.t. \(\sigma\):
L_1 = [(color; s_1) (size; s_1) (color; s_1)] \quad \text{(corresponding to A-D-F)}
L_2 = [(color; s_1) (size; s_2) (color; s_1)] \quad \text{(A-E-F)}
L_3 = [(size; s_1) (size; s_1) (color; s_1)] \quad \text{(B-D-F)}
L_4 = [(size; s_1) (size; s_2) (color; s_1)] \quad \text{(B-E-F)}
L_5 = [(color; s_2) (size; s_1) (color; s_1)] \quad \text{(C-D-F)}
L_6 = [(color; s_2) (size; s_2) (color; s_1)] \quad \text{(C-E-F)}

These are the only lists that span \(\sigma_1, \sigma_2, \) and \(\sigma_3\) w.r.t. \(\sigma\). From these, only \(L_4, L_5,\) and \(L_6\) do so properly. \(L_1\) does not span \(\sigma_1, \sigma_2,\) and \(\sigma_3\) properly (w.r.t. \(\sigma\)) because \([1 3]\) is a sublist of \([1 2 3]\) such that \([L_1(1) - L_1(3)]\), namely \([(\color{color}1) (\color{color}1) (\color{color}1)]\), is homogeneous and yet

\[
\sigma_1(\color{color}1) \cup \sigma_3(\color{color}1) = \{R, B\} \not\subset \sigma(\color{color}1) = \{\color{color}1, B\}.
\]

\(L_2\) fails for the same reason. For \(L_3, [1 2]\) is a sublist of \([1 2 3]\) such that

\[
[L_3(1) - L_3(2)] = [(\size{1} \size{1}) (\size{1})]
\]

is homogeneous but

\[
\sigma_1(\size{1}) \cup \sigma_2(\size{1}) = \{S, M, L\} \not\subset \sigma(\size{1}) = \{S, M, L\}.
\]

Accordingly, we have a total of three alternative extensions of \(\sigma\) w.r.t. \(\sigma_1, \sigma_2,\) and \(\sigma_3,\) corresponding to \(L_4, L_5,\) and \(L_6:\)

<table>
<thead>
<tr>
<th>(\sigma_4) (B-E-F)</th>
<th>(\sigma_5) (C-D-F)</th>
<th>(\sigma_6) (C-E-F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>color(\text{s}_1) = {\text{B}}</td>
<td>color(\text{s}_1) = {\text{B}}</td>
<td>color(\text{s}_1) = {\text{B}}</td>
</tr>
<tr>
<td>size(\text{s}_1) = {\text{L}}</td>
<td>size(\text{s}_1) = {\text{S}, \text{M}}</td>
<td>size(\text{s}_1) = {\text{S}, \text{M}, \text{L}}</td>
</tr>
<tr>
<td>color(\text{s}_2) = {\text{R}, \text{B}, \text{G}}</td>
<td>color(\text{s}_2) = {\text{R}}</td>
<td>color(\text{s}_2) = {\text{R}}</td>
</tr>
<tr>
<td>size(\text{s}_2) = {\text{M}}</td>
<td>size(\text{s}_2) = {\text{M}, \text{L}}</td>
<td>size(\text{s}_2) = {\text{M}}</td>
</tr>
</tbody>
</table>

so that

\[
\mathcal{AE}(\{\sigma_1, \sigma_2, \sigma_3\}, \sigma) = \{\sigma_4, \sigma_5, \sigma_6\}.
\]

Thus each alternative state is uniquely determined by the corresponding list that properly spans \(\sigma_1, \sigma_2,\) and \(\sigma_3\) w.r.t. \(\sigma\). Specifically, the ascription values of each alternative state are obtained by “flipping” (complementing) the corresponding ascription values of \(\sigma\) on the relevant coordinates of the respective spanning list. For coordinates (a.o. pairs) \((l; s)\) that are not in the spanning list \(L\), the original values of \(\sigma\) are retained, since in those cases we have \(\text{Pos}((l; s), L) = \emptyset\) and hence

\[
\sigma(l, s) \cup \bigcup_{i \in \text{Pos}((l; s), L)} \sigma_i(l, s) = \sigma(l, s).
\]

Intuitively, this ensures that every alternative state has a maximal disagreement with each extension of \(\sigma\).

We now depict the relevant states as ADFAs. First, the A DFA for the initial state \(\sigma\) is as follows:

Extensions \(\sigma_1, \sigma_2,\) and \(\sigma_3\) are shown in Figure 1.6. Finally, the ADFAs for the three computed extensions appear in Figure 1.7 As with the single-state case, the ADFAs for the alternative extensions must be constructed incrementally in order to ensure disjointness. As before, worlds are only added to a new A DFA if they are not already accepted by some previous A DFA. Applying this technique here would produce \(\sigma_4\) and \(\sigma_5\) as the two final alternative extensions, since \(\sigma_6\) is redundant: every world in it is accepted either by the \(\sigma_4\) A DFA or the \(\sigma_5\) A DFA.

\[\square\]
The following algorithm computes the set of lists that properly span states \( \sigma_1, \ldots, \sigma_m \) w.r.t. \( \sigma \):

1. Let \( \Phi \) be the set of all a.o. pairs for the system at hand. The size of this set will equal the power of the system.

2. Let \( \Psi \) be the set obtained from \( \Phi^m \) by filtering out all those lists \( [(l_1; s_1) \cdots (l_m; s_m)] \) for which

\[
\exists i \in \{1, \ldots, m\} . \sigma_i(l_i, s_i) = \sigma(l_i, s_i).
\]

That is,

\[
\Psi = \{[(l_1; s_1) \cdots (l_m; s_m)] \in \Phi^m | \sigma_i(l_i, s_i) \subset \sigma(l_i, s_i) \text{ for } i = 1, \ldots, m\}.
\]

Thus \( \Psi \) is the set of all and only those lists that span \( \sigma_1, \ldots, \sigma_m \) w.r.t. \( \sigma \).

3. From \( \Psi \), filter out those lists that do not properly span \( \sigma_1, \ldots, \sigma_m \) w.r.t. \( \sigma \), and return the result. To determine whether a list \( [(l_1; s_1) \cdots (l_m; s_m)] \) in \( \Psi \) properly spans \( \sigma_1, \ldots, \sigma_m \) w.r.t. \( \sigma \), do the following:

- Let \( f \) be a function that maps a.o. pairs to sets of positive integers. Initially, set \( f \leftarrow \lambda p . \emptyset \) for any a.o. pair \( p \).
- Let \( P \leftarrow \emptyset \).
- For \( i = 1, \ldots, m \):
  - \( f \leftarrow f[(l_i; s_i)] \rightarrow f(l_i, s_i) \cup \{i\} \);
  - \( P \leftarrow P \cup \{(l_i; s_i)\} \).
- For each pair \( (l; s) \in P \): if

\[
\bigcup_{i \in f(l; s)} \sigma_i(l, s) = \sigma
\]

return \textit{false}, else continue.

- Return \textit{true}.

With this algorithm, we can easily compute \( \text{AE}(\{\sigma_1, \ldots, \sigma_m\}, \sigma) \) as follows:

1. Let \( \Psi \) be the set of all and only those lists of a.o. pairs that properly span \( \sigma_1, \ldots, \sigma_m \) w.r.t. \( \sigma \).
2. Let $\Sigma \leftarrow \emptyset$.

3. For each list $L \in \Psi$:
   
   - Let $\sigma'$ be the unique state such that for any $l$ and $s$,
     \[
     \sigma'(l, s) = \sigma(l, s) \setminus \bigcup_{i \in Pos((l; s), L)} \sigma_i(l, s).
     \]
   
   - $\Sigma \leftarrow \Sigma \cup \{\sigma'\}$.

4. Return $\Sigma$.

The reader will verify that the more general definition of alternative state extensions subsumes the former notion in the following sense:

**Lemma 2:** $\text{Alt}(\sigma, \sigma', \sigma'')$ iff $\text{Alt}(\sigma, \{\sigma'\}, \sigma'')$.

The following result generalizes Lemma 1.

**Lemma 3:** If $\sigma_1, \ldots, \sigma_m, w \sqsubseteq \sigma$ and $w \not\sqsubseteq \sigma_i$ for every $i = 1, \ldots, m$, then there is some $\sigma' \sqsubseteq \sigma$ such that $\text{Alt}(\sigma, \{\sigma_1, \ldots, \sigma_m\}, \sigma')$ and $w \sqsubseteq \sigma'$.

**Proof:** By assumption, we have

\[
\forall l, s . w(l, s) \subseteq \sigma(l, s); \quad (1.12)
\]
\[
\forall i \in \{1, \ldots, m\} . \forall l, s . \sigma_i(l, s) \subseteq \sigma(l, s). \quad (1.13)
\]

Further, for each $i = 1, \ldots, m$ there is some a.o. pair $(l_i; s_i)$ such that

\[
w(l_i, s_i) \not\subseteq \sigma_i(l_i, s_i),
\]

meaning that there is some attribute value $\alpha_i$ such that

\[
w(l_i, s_i) = \{\alpha_i\} \quad (1.14)
\]
and
\[ \alpha_i \not\in \sigma_i(l_i, s_i). \] (1.15)

From (1.14) and (1.12) we infer
\[ \forall i \in \{1, \ldots, m\}. \alpha_i \in \sigma(l_i, s_i). \] (1.16)

Hence, from (1.15) and (1.13) we conclude
\[ \forall i \in \{1, \ldots, m\}. \sigma_i(l_i, s_i) \subset \sigma(l_i, s_i). \] (1.17)

Therefore, the list
\[ L = [(l_1; s_1) \cdots (l_m; s_m)] \]
spans \( \sigma_1, \ldots, \sigma_m \) w.r.t. \( \sigma \). Moreover, it does so properly. To see this, consider any sublist \([i_1 \cdots i_{m'}]\) of \([1 \cdots m]\) such that \([L(i_1) \cdots L(i_{m'})]\) is homogeneous, so that
\[ (l_{i_1}; s_{i_1}) = \cdots = (l_{i_{m'}}; s_{i_{m'}}) \]
and hence
\[ w(l_{i_1}, s_{i_1}) = \cdots = w(l_{i_{m'}}, s_{i_{m'}}), \]
which is to say
\[ \alpha_{i_1} = \cdots = \alpha_{i_{m'}}. \] (1.18)

Now suppose, by way of contradiction, that
\[ \bigcup_{j=1}^{m'} \sigma_{i_j}(l_{i_j}, s_{i_j}) = \sigma(l_{i_1}, s_{i_1}). \] (1.19)

From (1.14) and (1.12) we conclude
\[ w(l_{i_1}, s_{i_1}) = \{\alpha_{i_1}\} \subseteq \sigma(l_{i_1}, s_{i_1}), \]
so that
\[ \alpha_{i_1} \in \sigma(l_{i_1}, s_{i_1}). \] (1.20)

Hence, by (1.19),
\[ \alpha_{i_1} \in \bigcup_{j=1}^{m'} \sigma_{i_j}(l_{i_j}, s_{i_j}), \]
which means that there is some \( j \in \{1, \ldots, m'\} \) such that
\[ \alpha_{i_1} \in \sigma_{i_j}(l_{i_j}, s_{i_j}). \] (1.21)

From (1.15) we get \( \alpha_{i_1} \not\in \sigma_{i_j}(l_{i_j}, s_{i_j}). \) But, by (1.18), \( \alpha_{i_1} = \alpha_{i_j} \), hence \( \alpha_{i_1} \not\in \sigma_{i_j}(l_{i_j}, s_{i_j}), \) contradicting (1.21).

Therefore, \( L \) spans \( \sigma_1, \ldots, \sigma_m \) w.r.t. \( \sigma \) properly.

Now define \( \sigma' \supseteq \sigma \) as follows: for any \( l \) and \( s \),
\[ \sigma'(l, s) = \sigma(l, s) \setminus \bigcup_{i \in \text{Pos}(l; s), L} \sigma_i(l, s). \] (1.22)

By construction, \( \text{Alt}(\sigma, \{\sigma_1, \ldots, \sigma_m\}, \sigma') \). Further, we have \( w \subseteq \sigma' \). To prove this, we need to show that \( w(l, s) \subseteq \sigma'(l, s) \) for all \( l \) and \( s \). To that end, consider arbitrary \( l \) and \( s \). Either \( (l, s) \) occurs in \( L \) or not. If not, then \( \text{Pos}(l; s), L) = \emptyset \) and hence, from (1.22), \( \sigma'(l, s) = \sigma(l, s) \), so \( w(l, s) \subseteq \sigma'(l, s) \) follows from (1.12). Suppose, by contrast, that \( (l; s) \) occurs in \( L \), so that
\[ \text{Pos}(l; s), L) = \{i_1, \ldots, i_{m'}\} \]
for some \( m' \) such that \( 1 \leq m' \leq m \). From (1.15),
\[ \forall j \in \{1, \ldots, m'\}. \alpha_{i_j} \not\in \sigma_{i_j}(l_{i_j}, s_{i_j}). \] (1.23)
But

\[ \alpha_{i_1} = w(l_{i_1}, s_{i_1}), \ldots, \alpha_{i_{m'}} = w(l_{i_{m'}}, s_{i_{m'}}), \]

and since

\[ (l_{i_1}; s_{i_1}) = \cdots = (l_{i_{m'}}; s_{i_{m'}}) = (l; s), \]

we get \( \alpha_{i_1} = \cdots = \alpha_{i_{m'}}. \) Accordingly, \((1.23)\) yields

\[ \forall j \in \{1, \ldots, m'\} . \alpha_{i_1} \notin \sigma_{i_j}(l_{i_j}, s_{i_j}), \]

which, by virtue of \((1.24)\), becomes

\[ \forall j \in \{1, \ldots, m'\} . \alpha_{i_1} \notin \sigma_{i_j}(l, s). \]

It follows from \((1.25)\) that

\[ \alpha_{i_1} \notin \bigcup_{j=1}^{m'} \sigma_{i_j}(l, s), \]

or, equivalently,

\[ \alpha_{i_1} \notin \bigcup_{i \in \text{Pos}((l; s), L)} \sigma_{i}(l, s). \]

However,

\[ \alpha_{i_1} \in \sigma(l_{i_1}, s_{i_1}) = \sigma(l, s), \]

and hence we infer from \((1.26)\), \((1.27)\), and \((1.22)\) that

\[ \alpha_{i_1} \in \sigma'(l, s). \]

But, from \((1.24)\), \(w(l_{i_1}, s_{i_1}) = w(l, s)\), which is to say \(w(l, s) = \{\alpha_{i_1}\}\). We have thus shown that, in this case too, \(w(l, s) \subseteq \sigma'(l, s)\).

\[ \square \]

1.5 Interpreting first-order languages into system states

Consider a first-order vocabulary \( \Sigma = (C; R; V) \) consisting of a set of constant symbols \( C \); a set of relation symbols \( R \), with each \( R \in R \) having a unique positive arity; and a set of variables \( V \). An attribute interpretation of \( \Sigma \) into an attribute structure \( \mathcal{A} = (\{l_1 : A_1, \ldots, l_k : A_k\}; R) \) is a mapping \( I \) that assigns, to each relation symbol \( R \in R \) of arity \( n \):

1. a relation \( R^I \in R \) of some arity \( m \), called the realization of \( R \):

\[ R^I \subseteq A_{i_1} \times \cdots \times A_{i_m} \]

(where we might have \( m \neq n \)); and

2. a list of \( m \) pairs

\[ [(l_{i_1}; j_1) \cdots (l_{i_m}; j_m)] \]

called the profile of \( R \) and denoted by \( \text{Prof}(R) \), with \( 1 \leq j_x \leq n \) for each \( x = 1, \ldots, m \).

As will become clear soon, an attribute interpretation differs from a normal interpretation in that an atomic formula over system objects is “compiled” via profiles into an atomic formula over selected attribute values of (some of) those objects. The profile dictates which attributes of which objects will be selected. Accordingly, an atomic statement concerning system objects must be understood as an atomic statement concerning certain attribute values of those objects. Also note that unlike regular interpretations, an attribute interpretation does not fix the referents of the constant symbols. In Vivid, a constant symbol can dynamically come to denote an object during the course of a deduction (as more information is obtained about the diagram).

For the remainder of this section, fix a signature \( \Sigma = (C; R; V) \), an attribute structure

\[ \mathcal{A} = (\{l_1 : A_1, \ldots, l_k : A_k\}; R), \]

19
and an attribute interpretation $I$ of $\Sigma$ into $A$.

Suppose now that we are given an $A$-system $S = (\{s_1, \ldots, s_n\}; A)$. We define a constant assignment as a partial function $\rho$ from $C$ to $\{s_1, \ldots, s_n\}$; while a variable assignment is a total function $\chi$ from $V$ to $\{s_1, \ldots, s_n\}$. We write $\text{Dom}(\rho)$ for the domain of a constant assignment $\rho$, i.e., the set of all and only those constant symbols for which $\rho$ is defined. A total constant assignment will usually be written as $\widehat{\rho}$, with the hat indicating that the mapping is total. We will say that two constant assignments $\rho_1$ and $\rho_2$ have a conflict iff there is some $c \in \text{Dom}(\rho_1) \cap \text{Dom}(\rho_2)$ such that $\rho_1(c) \neq \rho_2(c)$. Therefore, if $\text{Dom}(\rho_1) \supseteq \text{Dom}(\rho_2)$ then $\rho_1$ and $\rho_2$ have a conflict iff $\rho_1 \not\supseteq \rho_2$.

Formulas $F$ over $\Sigma$ are defined as usual, with a term $t$ being either a variable or a constant symbol. We omit definitions of standard notions such as free variable occurrences, alphabetic equivalence, etc. The set of variables that occur free in a formula $F$ is denoted by $FV(F)$. We regard alphabetically equivalent formulas as identical. A sentence is a formula without any free variable occurrences. For any term $t$, we define $t^{\rho_X}$ as $\rho(c)$ if $t$ is a constant symbol $c$ and as $\chi(v)$ if $t$ is a variable $v$. Since $\rho$ is a partial function, $t^{\rho_X}$ may be undefined. We write $t^{\rho_X} \uparrow$ to indicate that $t^{\rho_X}$ is undefined, and $t^{\rho_X} \downarrow$ to indicate that it is defined.

By a named state we will mean a pair $(\sigma; \rho)$ consisting of a state $\sigma$ and a constant assignment $\rho$. We say that a named state $(\sigma'; \rho')$ is an extension of a named state $(\sigma; \rho)$, written $(\sigma'; \rho') \sqsubseteq (\sigma; \rho)$, iff $\sigma'$ is an extension of $\sigma$ (i.e., $\sigma' \subseteq \sigma$) and $\rho' \supseteq \rho$ (viewing the partial functions $\rho$ and $\rho'$ as sets of ordered pairs). Note that $\sqsubseteq$ is covariant on the state components but contravariant on the constant assignments. We say that $(\sigma'; \rho')$ is a proper extension of $(\sigma; \rho)$, written $(\sigma'; \rho') \subset (\sigma; \rho)$, iff $(\sigma'; \rho') \sqsubseteq (\sigma; \rho)$ and either $\sigma' \not\supseteq \sigma$ or $\rho' \not\supseteq \rho$. Further, $(\sigma'; \rho')$ is a finite extension of $(\sigma; \rho)$ iff $(\sigma'; \rho') \sqsubseteq (\sigma; \rho)$ and the difference $\rho' \setminus \rho$ is finite. We write $(\sigma'; \rho') \sqsupseteq (\sigma; \rho)$ (or $(\sigma'; \rho') \sqsubseteq (\sigma; \rho)$) to indicate that $(\sigma'; \rho')$ is a finite extension (respectively, a finite proper extension) of $(\sigma; \rho)$. A named state $(\sigma; \rho)$ is called a world if $\sigma$ is a world (every ascription of $\sigma$ is a valuation) and $\rho$ is total.22 As before, worlds do not have any extensions. If $(\sigma'; \rho') \sqsubseteq (\sigma; \rho)$ we might say that $(\sigma'; \rho')$ is obtainable from $(\sigma; \rho)$ by thinning, or conversely, that $(\sigma; \rho)$ is obtainable from $(\sigma'; \rho')$ by widening. By an assumption base $\beta$ we will mean a finite set of formulas. A context is a pair $\gamma = (\beta; (\sigma; \rho))$ consisting of an assumption base $\beta$ and a named state $(\sigma; \rho)$. Note that since the identity relation on each attribute is required to be decidable (by the computability proviso of Definition 1), the relation $\sqsubseteq$ is decidable as well.

**Lemma 4**: The relation $\sqsubseteq$ is a quasi-order on named states, i.e., it is reflexive and transitive.

We will now show how to assign a truth value—or an unknown token—to any formula $F$, given an arbitrary named state $(\sigma; \rho)$ (of an $A$-system $S = (\{s_1, \ldots, s_n\}; A)$) along with a variable assignment $\chi$. This is done by formally defining a mapping $I_{(\sigma; \rho)}/\chi$ from the set of all formulas to the three-element set

\[
\{\text{true}, \text{false}, \text{unknown}\}
\]

as follows. We begin by defining the truth value of a formula $F$ not with respect to an arbitrary state $\sigma$ (and assignments $\rho$ and $\chi$), but only with respect to a given world $w$ (as well as $\rho$ and $\chi$). This is denoted by $V^f_{(w; \rho)}/\chi[F]$. We will afterwards define $I_{(\sigma; \rho)}/\chi(F)$ in terms of $V^f_{(w; \rho)}/\chi[F]$ for $w \subseteq \sigma$.

First, the constants true and false are self-evaluating:

\[
V^f_{(w; \rho)}/\chi[\text{true}] = \text{true} \quad \text{and} \quad V^f_{(w; \rho)}/\chi[\text{false}] = \text{false}.
\]  

(1.29)

Next, consider an atomic formula $R(t_1, \ldots, t_n)$, where $R$ is a relation symbol of arity $n$ and profile

\[
[(l_{i_1}; j_1) \cdots (l_{i_m}; j_m)].
\]

We have:

\[
V^f_{(w; \rho)}/\chi[R^f(t_1, \ldots, t_n)] = \begin{cases} 
\text{unknown} & \text{if } \exists k \in \{1, \ldots, m\}. t^{\rho_X}_k \uparrow; \\
\text{true} & \text{if } R^f(w(l_{i_1}, t^{\rho_X}_{j_1}), \ldots, w(l_{i_m}, t^{\rho_X}_{j_m})); \\
\text{false} & \text{if } \neg R^f(w(l_{i_1}, t^{\rho_X}_{j_1}), \ldots, w(l_{i_m}, t^{\rho_X}_{j_m})). 
\end{cases}
\]

(1.30)

The semantics of the remaining connectives are given in accordance with the strong three-valued Kleene scheme, as shown in Figure 18.21 Conditionalis $F \Rightarrow G$ are construed as disjunctions $\neg F \lor G$, and biconditionals $F \Leftrightarrow G$ as conjunctions $(F \Rightarrow G) \land (G \Rightarrow F)$.

---

21See Arkoudas (2000) for details.

22This also overloads the term “world”: sometimes it refers to a state and sometimes to a named state. Again, the context will always disambiguate the use.
\[ \mathcal{V}^I_{(w; \rho) / \chi} \neg F = \begin{cases} 
true & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[F] = \text{false}; \\
false & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[F] = \text{true}; \\
\text{unknown} & \text{otherwise.} 
\end{cases} \tag{1.31} \]

\[ \mathcal{V}^I_{(w; \rho) / \chi}[F_1 \land F_2] = \begin{cases} 
true & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[F_1] = \text{true} \text{ and } \mathcal{V}^I_{(w; \rho) / \chi}[F_2] = \text{true}; \\
false & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[F_1] = \text{false} \text{ or } \mathcal{V}^I_{(w; \rho) / \chi}[F_2] = \text{false}; \\
\text{unknown} & \text{otherwise.} 
\end{cases} \tag{1.32} \]

\[ \mathcal{V}^I_{(w; \rho) / \chi}[F_1 \lor F_2] = \begin{cases} 
true & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[F_1] = \text{true} \text{ or } \mathcal{V}^I_{(w; \rho) / \chi}[F_2] = \text{true}; \\
false & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[F_1] = \text{false} \text{ and } \mathcal{V}^I_{(w; \rho) / \chi}[F_2] = \text{false}; \\
\text{unknown} & \text{otherwise.} 
\end{cases} \tag{1.33} \]

\[ \mathcal{V}^I_{(w; \rho) / \chi}[\forall v . F] = \begin{cases} 
true & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[v^{\sigma \rightarrow s_i}][F] = \text{true} \text{ for all } i \in \{1, \ldots, n\}; \\
false & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[v^{\sigma \rightarrow s_i}][F] = \text{false} \text{ for some } i \in \{1, \ldots, n\}; \\
\text{unknown} & \text{otherwise.} 
\end{cases} \tag{1.34} \]

\[ \mathcal{V}^I_{(w; \rho) / \chi}[\exists v . F] = \begin{cases} 
true & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[v^{\sigma \rightarrow s_i}][F] = \text{true} \text{ for some } i \in \{1, \ldots, n\}; \\
false & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[v^{\sigma \rightarrow s_i}][F] = \text{false} \text{ for all } i \in \{1, \ldots, n\}; \\
\text{unknown} & \text{otherwise.} 
\end{cases} \tag{1.35} \]

Figure 1.8: Strong Kleene semantics of complex formulas with respect to individual worlds.

Note that occurrences of symbols such as \( \forall \) and \( \exists \) on the right-hand sides of (1.30)—(1.35) occur as part of our metalanguage and should not be confused with object-level occurrences of these symbols in Vivid formulas. We will continue to use object-level symbols in different capacities without explicitly calling attention to the distinction; the context will always clarify the use.

We now define \( I_{(\sigma; \rho) / \chi}(F) \) as follows:

\[ I_{(\sigma; \rho) / \chi}(F) = \begin{cases} 
true & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[F] = \text{true} \text{ for every } w \subseteq \sigma; \\
false & \text{if } \mathcal{V}^I_{(w; \rho) / \chi}[F] = \text{false} \text{ for every } w \subseteq \sigma; \\
\text{unknown} & \text{otherwise.} 
\end{cases} \tag{1.36} \]

Example 11: Consider the signature \( \Sigma_1 = (C_{\text{clock}}; R_{\text{clock}}; V_{\text{clock}}) \) where the set of constant symbols is

\[ C_{\text{clock}} = \{ c_1, c_2, \ldots \} \]

the set of variables is \( V_{\text{clock}} = \{ x, y, z, x_1, y_1, z_1, \ldots \} \), and the set of relation symbols is

\[ R_{\text{clock}} = \{ \text{PM}, \text{AM}, \text{Ahead}, \text{Behind} \} \],

with \( \text{PM}, \text{AM} \) unary and \( \text{Ahead}, \text{Behind} \) binary.

Consider now the attribute structure

\[ \text{Clock} = \langle \text{hours} : \{0, \ldots, 23\}, \text{minutes} : \{0, \ldots, 59\}; \{ R_1, R_2, R_3, R_4 \} \rangle \]

where \( R_1 \subseteq \text{hours}, R_2 \subseteq \text{hours} \)

\[ R_3 \subseteq \text{hours} \times \text{minutes} \times \text{hours} \times \text{minutes}, \]

\[ R_4 \subseteq \text{hours} \times \text{minutes} \times \text{hours} \times \text{minutes} , \]

21
defined as follows: \( R_1(h) \Leftrightarrow h > 11, R_2(h) \Leftrightarrow h \leq 11, \)
\[
R_3(h_1, m_1, h_2, m_2) \Leftrightarrow h_1 > h_2 \lor (h_1 = h_2 \land m_1 > m_2),
\]
and
\[
R_4(h_1, m_1, h_2, m_2) \Leftrightarrow h_1 \leq h_2 \lor (h_1 = h_2 \land m_1 \leq m_2).
\]

We define an interpretation \( I \) of \( \Sigma_1 \) into this attribute structure by specifying a unique relation (in the structure) and a unique profile for each symbol in \( R_{\text{clock}} \). In particular, we set \( \text{PM} = R_1, \text{AM} = R_2, \text{Ahead} = R_3, \text{Behind} = R_4, \) and:
\[
\begin{align*}
\text{Prof}(\text{PM}) &= ([\text{hours}, 1]); \\
\text{Prof}(\text{AM}) &= ([\text{hours}, 1]); \\
\text{Prof}(\text{Ahead}) &= ([\text{hours}, 1], [\text{minutes}, 1], [\text{hours}, 2], [\text{minutes}, 2]); \\
\text{Prof}(\text{Behind}) &= ([\text{hours}, 1], [\text{minutes}, 1], [\text{hours}, 2], [\text{minutes}, 2]).
\end{align*}
\]

**Example 12:** Consider the system \( \{s_1, s_2\}; \text{Clock} \), where \( \text{Clock} \) is the attribute structure of Example 11. Let \( \sigma \) be the following state of this system:
\[
\begin{align*}
\text{hours}(s_1) &= \{9, 13\}, \\
\text{minutes}(s_1) &= 12, \\
\text{hours}(s_2) &= 8, \\
\text{minutes}(s_2) &= 27,
\end{align*}
\]
and let \( \rho \) be the partial constant assignment that maps \( c_1 \) to \( s_1 \) and \( c_2 \) to \( s_2 \). There are exactly two worlds \( w_1 \) and \( w_2 \) in \( \sigma \), where \( w_1(\text{hours}, s_1) = 9 \) and \( w_2(\text{hours}, s_1) = 13 \), with \( w_1 \) and \( w_2 \) in agreement everywhere else. We claim that the sentence \( \text{Ahead}(c_1, c_2) \) is true in \( (\sigma, \rho) \) for any variable assignment \( \chi \). Indeed, consider an arbitrary \( \chi \). Recalling the profile of \( \text{Ahead} \), definition (1.36) tells us that in order to have
\[
I_{(\sigma, \rho)/\chi}(\text{Ahead}(c_1, c_2)) = \text{true}
\]
we must have
\[
\nu^I_{(w; \rho)/\chi}[\text{Ahead}(c_1, c_2)] = \text{true}
\]
for every \( w \subseteq \sigma \). It is straightforward to verify that this is the case: We have
\[
\nu^I_{(w_1; \rho)/\chi}[\text{Ahead}(c_1, c_2)] = \text{true}
\]
because \( R_3(9, 12, 8, 27) \); and we also have
\[
\nu^I_{(w_2; \rho)/\chi}[\text{Ahead}(c_1, c_2)] = \text{true},
\]
given that \( R_3(13, 12, 8, 27) \).
\]

A couple of remarks are in order here. First, in practice, \( \nu^I_{(w; \rho)/\chi}[F] \) will almost always be \textbf{true} or \textbf{false}. There is only one way in which it can be \textbf{unknown}, and it is a rather uninteresting one, namely, if \( F \) contains constant symbols which are not in the domain of \( \rho \). Even then, the result might not be \textbf{unknown}. Consider, for instance, a formula such as \( R(c_1, c_2) \). If, as the first clause of (1.30) indicates, the profile of \( R \) involves only the first argument position, then \( \nu^I_{(w; \rho)/\chi}[R(c_1, c_2)] \) might return \textbf{true} or \textbf{false} even when \( c_2 \notin \text{Dom}(\rho) \). Alternatively, a conjunction \( F_1 \land F_2 \) (or disjunction \( F_1 \lor F_2 \)) might return \textbf{false} (respectively, \textbf{true}) even if \( F_2 \) contains undefined constant symbols, etc. But certainly as long as every constant symbol that appears in the formula \( F \) is covered by \( \rho \), the value \( \nu^I_{(w; \rho)/\chi}[F] \) will not be \textbf{unknown}. The second point, which will fully emerge in lemmas 11–14 is that the semantics are not fully compositional (functional) w.r.t. to \( (\sigma; \rho) \). That is, the value of \( I_{(\sigma, \rho)/\chi}(F) \) cannot be determined from the values of \( I_{(\sigma, \rho)/\chi}(F') \) for subformulas \( F' \) of \( F \). Nevertheless, the semantics are fully compositional when we take into account the values of the various subformulas in worlds \( w \subseteq \sigma \), as dictated by (1.36).

\footnote{Since \( \chi \) is always total, there is no issue of free variables of \( F \) not in the domain of \( \chi \).}
Lemma 5: If $\mathcal{V}^I_{(w;\rho)}/\chi[F] \neq \text{unknown}$ and $\rho \subseteq \rho'$, then $\mathcal{V}^I_{(\sigma;\rho')}/\chi[F] = \mathcal{V}^I_{(\sigma;\rho)}/\chi[F]$.

PROOF: A straightforward structural induction on $F$.

Lemma 6 (Thinning preserves truth values): If $(\sigma;\rho) \subseteq (\sigma';\rho')$ and

$$I_{(\sigma;\rho)}/\chi(F) \neq \text{unknown},$$

then $I_{(\sigma';\rho')}/\chi(F) = I_{(\sigma;\rho)}/\chi(F)$.

PROOF: Immediate by (1.36).

The following result is readily proved by induction on the structure of $F$. It is the three-valued-logic version of the standard coincidence theorem of universal algebra and logic, which states that two variable assignments that agree on the free variables of a formula $F$ are indistinguishable for the purposes of determining the truth value of $F$.

Lemma 7: If $\chi_1(v) = \chi_2(v)$ for every variable $v$ that has a free occurrence in $F$, then

$$\mathcal{V}_{(\sigma;\rho)}/\chi_1(F) = \mathcal{V}_{(\sigma;\rho)}/\chi_2(F).$$

PROOF: A straightforward induction on the structure of $F$.

Lemma 8: Let $R$ be a relation symbol of arity $n$ and profile $[(l_1; j_1) \ldots (l_m; j_m)]$, and suppose that $t_{jk}^{\rho_X}$ is defined for every $k = 1, \ldots, n$. Then:

(a) $I_{(\sigma;\rho)}/\chi(R(t_1, \ldots, t_n)) = \text{true}$ iff $\forall a_1 \in \sigma(l_1, t_{j_1}^{\rho_X}) \ldots \forall a_m \in \sigma(l_m, t_{j_m}^{\rho_X}). R^I(a_1, \ldots, a_m);

(b) $I_{(\sigma;\rho)}/\chi(R(t_1, \ldots, t_n)) = \text{false}$ iff $\forall a_1 \in \sigma(l_1, t_{j_1}^{\rho_X}) \ldots \forall a_m \in \sigma(l_m, t_{j_m}^{\rho_X}). \neg R^I(a_1, \ldots, a_m);

(c) $I_{(\sigma;\rho)}/\chi(R(t_1, \ldots, t_n)) = \text{unknown}$ iff $\exists a_1 \in \sigma(l_1, t_{j_1}^{\rho_X}) \ldots \exists a_m \in \sigma(l_m, t_{j_m}^{\rho_X}). R^I(a_1, \ldots, a_m)$ and $\exists a_1 \in \sigma(l_1, t_{j_1}^{\rho_X}) \ldots \exists a_m \in \sigma(l_m, t_{j_m}^{\rho_X}). \neg R^I(a_1, \ldots, a_m)$.

PROOF: We first observe that since the set $\sigma(l, s)$ is finite for every attribute $l$ and system object $s$, there exists a well-ordering $<_{\sigma(l,s)}$ for it. For one direction of part (a), assume $I_{(\sigma;\rho)}/\chi(R(t_1, \ldots, t_n)) = \text{true}$, so that

$$\forall w \subseteq \sigma. \mathcal{V}^I_{(w;\rho)}/\chi[R(t_1, \ldots, t_n)] = \text{true}.$$  \hspace{1cm} (1.37)

Define a world $w$ as follows:

$$w(l, s) = \begin{cases} a_k & \text{if } \exists k \in \{1, \ldots, m\}, (l, s) = (l_{a_k}, t_{j_k}^{\rho_X}); \\
\text{the } <_{\sigma(l,s)}-\text{smallest element of } \sigma(l, s) & \text{otherwise.}
\end{cases}$$

By (1.37), $\mathcal{V}^I_{(w;\rho)}/\chi[R(t_1, \ldots, t_n)] = \text{true}$, i.e., by (1.30),

$$R^I(w(l_1, t_{j_1}^{\rho_X}), \ldots, w(l_m, t_{j_m}^{\rho_X})).$$  \hspace{1cm} (1.38)

But, by construction, $w(l_{a_k}, t_{j_k}^{\rho_X}) = a_k$ for every $k = 1, \ldots, m$, so (1.38) is tantamount to $R^I(a_1, \ldots, a_m)$. Since $a_1, \ldots, a_m$ were chosen arbitrarily, it follows that

$$\forall a_1 \in \sigma(l_1, t_{j_1}^{\rho_X}) \ldots \forall a_m \in \sigma(l_m, t_{j_m}^{\rho_X}). R^I(a_1, \ldots, a_m).$$  \hspace{1cm} (1.39)

Conversely, suppose that (1.39) holds. Pick any world $w \subseteq \sigma$. From (1.39) and the fact that $w(l_{a_k}, t_{j_k}^{\rho_X}) \in \sigma(l_{a_k}, t_{j_k}^{\rho_X})$ for every $k = 1, \ldots, m$, we have

$$R^I(w(l_1, t_{j_1}^{\rho_X}), \ldots, w(l_m, t_{j_m}^{\rho_X})).$$

so, by (1.30),

$$\mathcal{V}^I_{(w;\rho)}/\chi[R(t_1, \ldots, t_n)] = \text{true}.$$ 

Since $w$ was chosen arbitrarily, it follows that

$$\forall w \subseteq \sigma. \mathcal{V}^I_{(w;\rho)}/\chi[R(t_1, \ldots, t_n)] = \text{true},$$

i.e., $I_{(\sigma;\rho)}/\chi(R(t_1, \ldots, t_n)) = \text{true}$. The proof of part (b) is identical to that of part (a). For part (c), suppose that $I_{(\sigma;\rho)}/\chi(R(t_1, \ldots, t_n)) = \text{unknown}$. There are two cases:
• (i) \( \exists w \subseteq \sigma : \forall_{(w; \rho)/\chi}[R(t_1, \ldots, t_n)] = \text{unknown} \); or

• (ii) \( \exists w_1 \subseteq \sigma , w_2 \subseteq \sigma : \forall_{(w_1; \rho)/\chi}[R(t_1, \ldots, t_n)] = \text{true} \) and \( \forall_{(w_2; \rho)/\chi}[R(t_1, \ldots, t_n)] = \text{false} \).

The first case is impossible because, by (1.30), an identity of the form
\[
\forall_{(w_1; \rho)/\chi}[R(t_1, \ldots, t_n)] = \text{unknown}
\]
can only obtain if \( \exists k \in \{1, \ldots, m\} : t_{j_k}^{\rho \chi} \uparrow \). However, we have assumed that \( t_{j_k}^{\rho \chi} \) is defined for every \( k = 1, \ldots, m \). Accordingly, (ii) must be the case, i.e., for some \( w_1 \subseteq \sigma \) and \( w_2 \subseteq \sigma \) we have
\[
R_I(w_1(l_{i_1}, t_{j_1}^{\rho \chi}), \ldots, w_1(l_{i_m}, t_{j_m}^{\rho \chi}))
\]
and
\[
R_I(w_2(l_{i_1}, t_{j_1}^{\rho \chi}), \ldots, w_2(l_{i_m}, t_{j_m}^{\rho \chi})),
\]
and the result now follows immediately since \( w_1(l_{i_k}, t_{j_k}^{\rho \chi}) \in \sigma(l_{i_k}, t_{j_k}^{\rho \chi}) \) for every \( k = 1, \ldots, m \), and likewise for \( w_2(l_{i_k}, t_{j_k}^{\rho \chi}) \). Conversely, suppose that
\[
R_I(a_1, \ldots, a_m)
\]
and
\[
R_I(a'_1, \ldots, a'_m),
\]
where \( a_k, a'_k \in \sigma(l_{i_k}, t_{j_k}^{\rho \chi}) \) for \( k = 1, \ldots, m \). Then, by the same technique that was employed in the proof of part (a), i.e., by using the well-ordering \( <_{\sigma(l,s)} \), we can define worlds \( w, w' \subseteq \sigma \) such that
\[
\forall k \in \{1, \ldots, m\} : w(l_{i_k}, t_{j_k}^{\rho \chi}) = a_k \quad \text{and} \quad w'(l_{i_k}, t_{j_k}^{\rho \chi}) = a'_k.
\]
Therefore, (1.42) and (1.43) become tantamount, respectively, to
\[
R_I(w(l_{i_1}, t_{j_1}^{\rho \chi}), \ldots, w(l_{i_m}, t_{j_m}^{\rho \chi}))
\]
and
\[
R_I(w'(l_{i_1}, t_{j_1}^{\rho \chi}), \ldots, w'(l_{i_m}, t_{j_m}^{\rho \chi})).
\]
Consequently, \( \forall_{(w; \rho)/\chi}[R(t_1, \ldots, t_n)] = \text{true} \) and
\[
\forall_{(w'; \rho)/\chi}[R(t_1, \ldots, t_n)] = \text{false},
\]
and thus \( I_{(\sigma; \rho)/\chi}(R(t_1, \ldots, t_n)) = \text{unknown} \).

The following is a direct consequence of the finite size of the ascription values, the finite number of system objects, and Lemma 7.

**Lemma 9:** \( I_{(\sigma; \rho)/\chi} \) is computable for any named state \( (\sigma; \rho) \) and variable assignment \( \chi \).

**Lemma 10:** (a) \( I_{(\sigma; \rho)/\chi}(\neg F) = \text{true} \) iff \( I_{(\sigma; \rho)/\chi}(F) = \text{false} \); and (b) \( I_{(\sigma; \rho)/\chi}(\neg F) = \text{false} \) iff \( I_{(\sigma; \rho)/\chi}(F) = \text{true} \). Therefore,
\[
I_{(\sigma; \rho)/\chi}(\neg F) = \text{unknown} \iff I_{(\sigma; \rho)/\chi}(F) = \text{unknown}.
\]

**Proof:** Assume \( I_{(\sigma; \rho)/\chi}(\neg F) = \text{true} \), so that
\[
\forall w \subseteq \sigma : \forall_{(w; \rho)/\chi}[\neg F] = \text{true}.
\]
From (1.44) and (1.31) it follows that
\[
\forall w \subseteq \sigma : \forall_{(w; \rho)/\chi}[F] = \text{false},
\]
so now from \(1.45\) and \(1.36\) we conclude
\[I_{(\sigma; \rho)/\chi}(F) = \text{false}.\]

Conversely, assume \(I_{(\sigma; \rho)/\chi}(F) = \text{false}\), so that
\[\forall w \subseteq \sigma . \; \mathcal{V}_{(w; \rho)/\chi}[F] = \text{false}.\] (1.46)

From \(1.46\) and \(1.31\) it follows that
\[\forall w \subseteq \sigma . \; \mathcal{V}_{(w; \rho)/\chi}[-F] = \text{true},\]
which is to say \(I_{(\sigma; \rho)/\chi}(-F) = \text{true}\). Part (b) follows similarly.

The above lemma shows that the truth table for negation in Vivid is as follows:

<table>
<thead>
<tr>
<th>(I_{(\sigma; \rho)/\chi}(F))</th>
<th>(I_{(\sigma; \rho)/\chi}(\neg F))</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
</tr>
</tbody>
</table>

It coincides completely with the standard Kleene semantics.

**Lemma 11:** (a) \(I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{true} \iff I_{(\sigma; \rho)/\chi}(F_1) = \text{true} \; \text{and} \; I_{(\sigma; \rho)/\chi}(F_2) = \text{true}\).
(b) If \(I_{(\sigma; \rho)/\chi}(F_1) = \text{false} \; \text{or} \; I_{(\sigma; \rho)/\chi}(F_2) = \text{false} \; \text{then} \; I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{false}. \) However, the converse is not true. In particular, we may have:
\[I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{false}, \; I_{(\sigma; \rho)/\chi}(F_1) = \text{unknown}, \; I_{(\sigma; \rho)/\chi}(F_2) = \text{unknown}.
(c) If \(\{I_{(\sigma; \rho)/\chi}(F_1), I_{(\sigma; \rho)/\chi}(F_2)\} = \{\text{true, unknown}\} \; \text{then} \; I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{unknown}.
(d) If \(I_{(\sigma; \rho)/\chi}(F_1) = \text{unknown} \; \text{and} \; I_{(\sigma; \rho)/\chi}(F_2) = \text{unknown} \; \text{then} \; I_{(\sigma; \rho)/\chi}(F_1 \land F_2) \neq \text{true}.

**PROOF:** (a) Assume \(I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{true}\), so that
\[\forall w \subseteq \sigma . \; \mathcal{V}_{(w; \rho)/\chi}[F_1 \land F_2] = \text{true}.\] (1.47)

From \(1.32\) and \(1.47\) we infer
\[\forall w \subseteq \sigma . \; \mathcal{V}_{(w; \rho)/\chi}[F_1] = \text{true}\] (1.48)

and
\[\forall w \subseteq \sigma . \; \mathcal{V}_{(w; \rho)/\chi}[F_2] = \text{true},\] (1.49)
i.e., \(I_{(\sigma; \rho)/\chi}(F_1) = \text{true} \; \text{and} \; I_{(\sigma; \rho)/\chi}(F_2) = \text{true}\). Conversely, suppose that \(I_{(\sigma; \rho)/\chi}(F_1) = \text{true} \; \text{and} \; I_{(\sigma; \rho)/\chi}(F_2) = \text{true}\), so that \(1.48\) and \(1.49\) hold. Pick any \(w' \subsetneq \sigma\). From \(1.48\) and \(1.49\) we obtain \(\mathcal{V}_{(w'; \rho)/\chi}[F_1] = \text{true}\) and \(\mathcal{V}_{(w'; \rho)/\chi}[F_2] = \text{true}\), hence \(\mathcal{V}_{(w'; \rho)/\chi}[F_1 \land F_2] = \text{true}\). Since \(w'\) was chosen arbitrarily, it follows that
\[\forall w \subseteq \sigma . \; \mathcal{V}_{(w; \rho)/\chi}[F_1 \land F_2] = \text{true},\]
which is to say \(I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{true}\).

(b) Suppose that \(I_{(\sigma; \rho)/\chi}(F_1) = \text{false} \; \text{or} \; I_{(\sigma; \rho)/\chi}(F_2) = \text{false}\. Without loss of generality, assume the former, so that
\[\forall w \subseteq \sigma . \; \mathcal{V}_{(w; \rho)/\chi}[F_1] = \text{false}.\] (1.50)

Then, by \(1.32\),
\[\forall w \subseteq \sigma . \; \mathcal{V}_{(w; \rho)/\chi}[F_1 \land F_2] = \text{false},\]
i.e., \(I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{false}\). To see that the converse does not hold, refer to the second part of Example \([12]\) and let \(F\) be the sentence \(PM(c_1)\). Although we have \(I_{(\sigma; \rho)/\chi}(F) = \text{unknown}\) and \(I_{(\sigma; \rho)/\chi}(-F) = \text{unknown}\), we have \(I_{(\sigma; \rho)/\chi}(F \land -F) = \text{false}\) because in every world \(w \subseteq \sigma\), \(\mathcal{V}_{(w; \rho)/\chi}[F \land -F] = \text{false}\). This is a key difference from
Immediate from part (a).

Therefore, by (1.36),

\[ I_{(\sigma; \rho)/\chi}(F_1) = \text{true} \]  \hspace{1cm} (1.51)

and

\[ I_{(\sigma; \rho)/\chi}(F_2) = \text{unknown}, \]  \hspace{1cm} (1.52)

so that

\[ \forall w \subseteq \sigma \cdot V_{(w; \rho)/\chi}^I[F_1] = \text{true}. \]  \hspace{1cm} (1.53)

On the basis of (1.52) and (1.36), we distinguish the following two cases (which are jointly exhaustive, though not mutually exclusive):

\[ \exists w \subseteq \sigma \cdot V_{(w; \rho)/\chi}^I[F_2] = \text{unknown}; \]  \hspace{1cm} (1.54)

or

\[ \exists w_1 \subseteq \sigma \cdot \exists w_2 \subseteq \sigma \cdot V_{(w_1; \rho)/\chi}^I[F_2] = \text{true} \quad \text{and} \quad V_{(w_2; \rho)/\chi}^I[F_2] = \text{false}. \]  \hspace{1cm} (1.55)

Suppose (1.54) is the case, and let \( w' \subseteq \sigma \) be the world whose existence is asserted by (1.54). Then

\[ V_{(w'; \rho)/\chi}^I[F_2] = \text{unknown}, \]

while, from (1.53), \( V_{(w'; \rho)/\chi}^I[F_1] = \text{true} \). Therefore, by (1.32),

\[ V_{(w'; \rho)/\chi}^I[F_1 \land F_2] = \text{unknown}, \]

and hence, by (1.36), \( I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{unknown} \). Next, suppose that (1.55) is the case, and let \( w_1 \) and \( w_2 \) be the worlds whose existence is asserted by (1.55). Then, by (1.53),

\[ V_{(w_1; \rho)/\chi}^I[F_1 \land F_2] = \text{true} \quad \text{and} \quad V_{(w_2; \rho)/\chi}^I[F_1 \land F_2] = \text{false}. \]

Therefore, by (1.36), \( I_{(\sigma; \rho)/\chi}(F_1 \land F_2) = \text{unknown} \).

(d) Immediate from part (a).

The foregoing lemma establishes that the truth table for conjunctions in Vivid, with \((\sigma; \rho)\) fixed, is as shown in Figure 1.9.

The following result is the dual of Lemma 11, both in content and in proof.

**Lemma 12:** (a) \( I_{(\sigma; \rho)/\chi}(F_1 \lor F_2) = \text{false} \) if \( I_{(\sigma; \rho)/\chi}(F_1) = \text{false} \) and \( I_{(\sigma; \rho)/\chi}(F_2) = \text{false} \).

(b) If \( I_{(\sigma; \rho)/\chi}(F_1) = \text{true} \) or \( I_{(\sigma; \rho)/\chi}(F_2) = \text{true} \) then \( I_{(\sigma; \rho)/\chi}(F_1 \lor F_2) = \text{true} \). However, the converse is not true.

In particular, we may have:

\[ I_{(\sigma; \rho)/\chi}(F_1 \lor F_2) = \text{true}, I_{(\sigma; \rho)/\chi}(F_1) = \text{unknown}, I_{(\sigma; \rho)/\chi}(F_2) = \text{unknown}. \]
(c) If \( \{I_{(σ, ρ)/χ}(F_1), I_{(σ, ρ)/χ}(F_2)\} = \{\text{false, unknown}\} \) then \( I_{(σ, ρ)/χ}(F_1 ∨ F_2) = \text{unknown} \).

(d) If \( I_{(σ, ρ)/χ}(F_1) = \text{unknown} \) and \( I_{(σ, ρ)/χ}(F_2) = \text{unknown} \) then \( I_{(σ, ρ)/χ}(F_1 ∨ F_2) \neq \text{false} \).

**PROOF:**

(a) Assume \( I_{(σ, ρ)/χ}(F_1 ∨ F_2) = \text{false} \), so that

\[
∀ w \subseteq σ . \mathcal{V}^I_{(w; ρ)/χ}[F_1 ∨ F_2] = \text{false}.
\]

From (1.33) and (1.56) we infer

\[
∀ w \subseteq σ . \mathcal{V}^I_{(w; ρ)/χ}[F_1] = \text{false}
\]

and

\[
∀ w \subseteq σ . \mathcal{V}^I_{(w; ρ)/χ}[F_2] = \text{false},
\]

i.e., \( I_{(σ, ρ)/χ}(F_1) = \text{false} \) and \( I_{(σ, ρ)/χ}(F_2) = \text{false} \). Conversely, suppose that \( I_{(σ, ρ)/χ}(F_1) = \text{false} \) and

\[
I_{(σ, ρ)/χ}(F_2) = \text{false},
\]

so that (1.57) and (1.58) hold. Pick any \( w' \subseteq σ \). From (1.57) and (1.58) we obtain \( \mathcal{V}^I_{(w'; ρ)/χ}[F_1] = \text{false} \) and \( \mathcal{V}^I_{(w'; ρ)/χ}[F_2] = \text{false} \), hence \( \mathcal{V}^I_{(w'; ρ)/χ}[F_1 ∨ F_2] = \text{false} \). Since \( w' \) was chosen arbitrarily, it follows that

\[
∀ w \subseteq σ . \mathcal{V}^I_{(w; ρ)/χ}[F_1 ∨ F_2] = \text{false},
\]

which is to say \( I_{(σ, ρ)/χ}(F_1 ∨ F_2) = \text{false} \).

(b) Suppose that \( I_{(σ, ρ)/χ}(F_1) = \text{true} \) or \( I_{(σ, ρ)/χ}(F_2) = \text{true} \). Without loss of generality, assume the former, so that

\[
∀ w \subseteq σ . \mathcal{V}^I_{(w; ρ)/χ}[F_1] = \text{true}.
\]

Then, by (1.33),

\[
∀ w \subseteq σ . \mathcal{V}^I_{(w; ρ)/χ}[F_1 ∨ F_2] = \text{true},
\]

i.e., \( I_{(σ, ρ)/χ}(F_1 ∨ F_2) = \text{true} \). To see that the converse does not hold, refer again to the second part of Example 12, and again let \( F \) be the sentence \( P(\cdot) \). Although we have \( I_{(σ, ρ)/χ}(F) = \text{unknown} \) and \( I_{(σ, ρ)/χ}(¬F) = \text{unknown} \), we have \( I_{(σ, ρ)/χ}(F ∨ ¬F) = \text{true} \) because in every world \( w \subseteq σ \), \( \mathcal{V}^I_{(w; ρ)/χ}[F ∨ ¬F] = \text{true} \).

(c) Without loss of generality, suppose that

\[
I_{(σ, ρ)/χ}(F_1) = \text{false}
\]

and

\[
I_{(σ, ρ)/χ}(F_2) = \text{unknown},
\]

so that

\[
∀ w \subseteq σ . \mathcal{V}^I_{(w; ρ)/χ}[F_1] = \text{false}.
\]

On the basis of (1.61) and (1.36), we distinguish the following two cases (jointly exhaustive, not necessarily mutually exclusive):

\[
∃ w \subseteq σ . \mathcal{V}^I_{(w; ρ)/χ}[F_2] = \text{unknown};
\]

or

\[
∃ w_1 \subseteq σ . ∃ w_2 \subseteq σ . \mathcal{V}^I_{(w_1; ρ)/χ}[F_2] = \text{true} \quad \text{and} \quad \mathcal{V}^I_{(w_2; ρ)/χ}[F_2] = \text{false}.
\]

Suppose (1.63) is the case, and let \( w' \subseteq σ \) be the world whose existence is asserted by (1.63). Then

\[
\mathcal{V}^I_{(w'; ρ)/χ}[F_2] = \text{unknown},
\]

while, from (1.62), \( \mathcal{V}^I_{(w'; ρ)/χ}[F_1] = \text{false} \). Therefore, by (1.33),

\[
\mathcal{V}^I_{(w'; ρ)/χ}[F_1 ∨ F_2] = \text{unknown},
\]

and hence, by (1.36), \( I_{(σ, ρ)/χ}(F_1 ∨ F_2) = \text{unknown} \). Next, suppose that (1.64) is the case, and let \( w_1 \) and \( w_2 \) be the worlds whose existence is asserted by (1.64). Then, by (1.62),

\[
\mathcal{V}^I_{(w_1; ρ)/χ}[F_1 ∨ F_2] = \text{true} \quad \text{and} \quad \mathcal{V}^I_{(w_2; ρ)/χ}[F_1 ∨ F_2] = \text{false}.
\]

Therefore, by (1.36), \( I_{(σ, ρ)/χ}(F_1 ∨ F_2) = \text{unknown} \).

(d) Immediate from (a).
Figure 1.10: Semantics of disjunctions in Vivid. The difference from Kleene semantics (weak and strong) lies in the last row.

Lemma 13: 
(a) \( I_{(\sigma; \rho) / \chi}(\forall v . F) = \text{true} \) iff \( I_{(\sigma; \rho) / \chi[v \mapsto s_i]}(F) = \text{true} \) for every \( i = 1, \ldots, n \).
(b) If \( I_{(\sigma; \rho) / \chi[v \mapsto s_i]}(F) = \text{false} \) for some system object \( s_i \), then \( I_{(\sigma; \rho) / \chi}(\forall v . F) = \text{false} \). However, the converse does not hold. In particular, we may have \( I_{(\sigma; \rho) / \chi}[v \mapsto s_i](F) \neq \text{false} \) for every \( i = 1, \ldots, n \).

PROOF: Assume \( I_{(\sigma; \rho) / \chi}(\forall v . F) = \text{true} \), so that 
\[
\forall w \sqsubseteq \sigma . \mathcal{V}_{(w; \rho) / \chi}[\forall v . F] = \text{true}. \tag{1.65}
\]
Pick any \( i = 1, \ldots, n \), and any \( w \sqsubseteq \sigma \). From (1.65), we get 
\[
\mathcal{V}_{(w; \rho) / \chi}(\forall v . F) = \text{true},
\]
so, by (1.34), \( \mathcal{V}_{(w; \rho) / \chi[v \mapsto s_i]}[F] = \text{true} \). Since \( w \) was chosen arbitrarily, we may infer 
\[
\forall w \sqsubseteq \sigma . \mathcal{V}_{(w; \rho) / \chi[v \mapsto s_i]}[F] = \text{true},
\]
i.e., 
\[
I_{(\sigma; \rho) / \chi[v \mapsto s_i]}(F) = \text{true}.
\]
Since \( i \) was chosen arbitrarily, we conclude 
\[
\forall i \in \{1, \ldots, n\} . \forall w \sqsubseteq \sigma . \mathcal{V}_{(w; \rho) / \chi}[\forall v . F] = \text{true}. \tag{1.66}
\]
Conversely, suppose that (1.66) holds. Pick any \( w \sqsubseteq \sigma \), and any \( i \in \{1, \ldots, n\} \). From (1.66) we get 
\[
I_{(\sigma; \rho) / \chi[v \mapsto s_i]}(F) = \text{true},
\]
and therefore 
\[
\mathcal{V}_{(w; \rho) / \chi[v \mapsto s_i]}[F] = \text{true}.
\]
Since \( i \) was chosen arbitrarily, 
\[
\forall i \in \{1, \ldots, n\} . \mathcal{V}_{(w; \rho) / \chi[v \mapsto s_i]}[F] = \text{true},
\]
and thus, by (1.34), 
\[
\mathcal{V}_{(w; \rho) / \chi}[\forall v . F] = \text{true}.
\]
Since \( w \) was chosen arbitrarily, we conclude: 
\[
\forall w \sqsubseteq \sigma . \mathcal{V}_{(w; \rho) / \chi}[\forall v . F] = \text{true},
\]
i.e., \( I_{(\sigma; \rho) / \chi}(\forall v . F) = \text{true} \). More succinctly:
The problem lies in the quantifier swap. Existential and universal quantifiers may not be arbitrarily transposed, since there is usually a logical deduction enclosing the existential instantiation (in this case, the universal generalization that began with “Pick any witness for an existential instantiation, and therefore it cannot appear as a free variable in the conclusion of a larger inference.”)

The error is elementary but subtle nevertheless—it occurs in the derivation of (1.68). The variable \( i \) was introduced as a witness for an existential instantiation, and therefore it cannot appear as a free variable in the conclusion of a larger deduction enclosing the existential instantiation (in this case, the universal generalization that began with “Pick any witness for an existential instantiation, and therefore it cannot appear as a free variable in the conclusion of a larger inference.”) immediately after (1.67)\(^{24}\).

To establish that the converse does not actually hold, consider the system \((\{{s_1, s_2, s_3}\}; A_R)\), where \( A_R \) is the map-coloring attribute structure of Section 1.10 and let \( \sigma \) be a state such that:

\[
\begin{align*}
\sigma(\text{color}, s_1) &= \{G, R\}, \\
\sigma(\text{color}, s_2) &= \{R\}, \\
\sigma(\text{color}, s_3) &= \{G\},
\end{align*}
\]

where all three regions are adjacent to one another\(^{25}\). Diagrammatically, we have:

\[I_{(\sigma; \rho)}(\forall v . F) = \text{false} \quad \iff \quad (\text{by (1.34)})\]

\[\forall w \subseteq \sigma . I_{(\rho; \sigma)}(\forall v . F) = \text{true} \quad \iff \quad (\text{by (1.36)})\]

\[\forall w \subseteq \sigma . \forall i \in \{1, \ldots, n\} . I_{(\rho; \sigma)}(\forall v . F) = \text{true} \quad \iff \quad (\text{quantifier swap})\]

\[\forall i \in \{1, \ldots, n\} . \forall w \subseteq \sigma . I_{(\rho; \sigma)}(\forall v . F) = \text{true} \quad \iff \quad (\text{by (1.36)})\]

For the first half of part (b), assume:

\[I_{(\sigma; \rho)}(\forall v . F) = \text{false}\]

for some \( i \in \{1, \ldots, n\} \), so that:

\[\forall w \subseteq \sigma . I_{(\rho; \sigma)}(\forall v . F) = \text{true}\]

Hence,

\[\forall w \subseteq \sigma . \forall i \in \{1, \ldots, n\} . I_{(\rho; \sigma)}(\forall v . F) = \text{true}\]

which is to say \( I_{(\sigma; \rho)}(\forall v . F) = \text{false} \).

Incidentally, it is remarkably easy to arrive at an erroneous proof of the converse, and it is instructive to briefly examine an attempt at such a proof:

Assume \( I_{(\sigma; \rho)}(\forall v . F) = \text{false} \), so that:

\[\forall w \subseteq \sigma . \forall i \in \{1, \ldots, n\} . I_{(\rho; \sigma)}(\forall v . F) = \text{true}\]

(1.67)

Pick any \( w \subseteq \sigma \). From (1.67), we infer:

\[\forall i \in \{1, \ldots, n\} . I_{(\rho; \sigma)}(\forall v . F) = \text{false}\]

hence, by (1.34),

\[\forall i \in \{1, \ldots, n\} . I_{(\rho; \sigma)}(\forall v . F) = \text{false}\]

(1.68)

The error is elementary but subtle nevertheless—it occurs in the derivation of (1.68). The variable \( i \) was introduced as a witness for an existential instantiation, and therefore it cannot appear as a free variable in the conclusion of a larger deduction enclosing the existential instantiation (in this case, the universal generalization that began with “Pick any witness for an existential instantiation, and therefore it cannot appear as a free variable in the conclusion of a larger inference.”) immediately after (1.67)\(^{24}\).

The problem lies in the quantifier swap. Existential and universal quantifiers may not be arbitrarily transposed, since there is usually a logical dependency between the two variables.

\(^{24}\)The error is more glaring in a more succinct version of this “proof”:

\[I_{(\sigma; \rho)}(\forall v . F) = \text{false} \quad \Rightarrow \quad (\text{by (1.34)})\]

\[\forall w \subseteq \sigma . I_{(\rho; \sigma)}(\forall v . F) = \text{false} \quad \Rightarrow \quad (\text{by (1.34)})\]

\[\forall w \subseteq \sigma . \exists i \in \{1, \ldots, n\} . I_{(\rho; \sigma)}(\forall v . F) = \text{false} \quad \Rightarrow \quad (\text{quantifier swap})\]

\[\exists i \in \{1, \ldots, n\} . \forall w \subseteq \sigma . I_{(\rho; \sigma)}(\forall v . F) = \text{false} \quad \Rightarrow \quad (\text{by (1.34)})\]

\[\exists i \in \{1, \ldots, n\} . \forall w \subseteq \sigma . I_{(\rho; \sigma)}(\forall v . F) = \text{false} \quad \Rightarrow \quad (\text{by (1.36)})\]

\(^{25}\)I.e., \( \sigma(\text{neighbors}, s_i) = \{s_1, s_2, s_3\} \) for every \( s_i \).
Thus $\sigma$ contains exactly two worlds, $w_1$ and $w_2$:

<table>
<thead>
<tr>
<th></th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$:</td>
<td>${G, R}$</td>
<td>$R$</td>
<td>$G$</td>
</tr>
</tbody>
</table>

Further, let $\rho$ be the partial assignment that maps the constant symbol $r_1$ to $s_1$, let $\chi$ be an arbitrary variable assignment, and let $F$ be the following formula:

\[
\forall u. \text{adjacent}(u, r_1) \land u \neq r_1 \Rightarrow \neg \text{sameColor}(u, r_1).
\]

This simply says that every region adjacent to $r_1$ other than itself has a different color from it. Now this is false, because it cannot be satisfied no matter how we assign colors. That is, in every world (in this case, both in $w_1$ and in $w_2$), $F$ comes out false. In $w_1$, $r_1$ (i.e., $s_1$) is colored green and thus it clashes with $s_3$; in $w_2$, it is colored red and thus it clashes with $s_2$:

\[
I_{(\sigma; \rho)/\chi}[F] = \text{false},
\]

and

\[
I_{(\sigma; \rho)/\chi}[F] = \text{false},
\]

(where $I$ is the obvious interpretation; see Section 1.10 for details). And yet for every system object $s_i$ we have

\[
I_{(\sigma; \rho)/\chi[u \mapsto s_i]}(\text{adjacent}(u, r_1) \land u \neq r_1 \Rightarrow \neg \text{sameColor}(u, r_1)) \neq \text{false}.
\]

In particular, and letting $M$ stand for the body of $F$ for brevity:

\[
M = \text{adjacent}(u, r_1) \land u \neq r_1 \Rightarrow \neg \text{sameColor}(u, r_1),
\]

we have:

\[
I_{(\sigma; \rho)/\chi[u \mapsto s_1]}(M) = \text{true}; \quad (1.69)
\]

\[
I_{(\sigma; \rho)/\chi[u \mapsto s_2]}(M) = \text{unknown}; \quad (1.70)
\]

\[
I_{(\sigma; \rho)/\chi[u \mapsto s_3]}(M) = \text{unknown}. \quad (1.71)
\]

(1.69) holds because $M$ is vacuously true when $u$ refers to the same region as $r_1$ (namely, $s_1$), since the antecedent is then false. (1.70) holds because the body comes out true under one world and false under another:

\[
I_{(\sigma; \rho)/\chi[u \mapsto s_2]}[M] = \text{true};
\]

\[
I_{(\sigma; \rho)/\chi[u \mapsto s_3]}[M] = \text{true}.
\]

Likewise for (1.71):

\[
I_{(\sigma; \rho)/\chi[u \mapsto s_3]}[M] = \text{false};
\]

\[
I_{(\sigma; \rho)/\chi[u \mapsto s_3]}[M] = \text{true}.
\]

Accordingly, Kleene’s three-value semantics would entail the counter-intuitive identity $I_{(\sigma; \rho)/\chi}(F) = \text{unknown}$, even though it is obvious that $F$ comes out false in $(\sigma; \rho)$ and $\chi$. This is the major point of departure between the semantics of Vivid formulas, as given by (1.36), and Kleene’s overly conservative three-valued logic.

The fact that universal and existential generalizations have the same logical structure as conjunctions and disjunctions, respectively, accounts for the similarity between Lemma 13 and Lemma 11 whose contents are essentially identical. The only ostensible difference between the two is part (c) of Lemma 11 which claims that the conjunction of true and unknown components results in unknown, and which has no direct analogue in Lemma 13. Indeed, the counterexample in the proof of Lemma 13 depicts a situation where every instance of a universal generalization is either true or unknown and yet the generalization itself is false: $I_{(\sigma; \rho)/\chi}(\forall v . F) = \text{false}$ even though $I_{(\sigma; \rho)/\chi[v\mapsto s_i]}(F) \neq \text{false}$ for all $i$. The disparity here is only apparent. The issue is that conjunctions are always binary whereas universal quantifications may have an arbitrarily large number of instances. In fact there is an analogue of part (c) for universal generalizations, namely: If there are only two system objects $s_1$ and $s_2$, and $I_{(\sigma; \rho)/\chi[v\mapsto s_1]}(F) = \text{true}$ while $I_{(\sigma; \rho)/\chi[v\mapsto s_2]}(F) = \text{unknown}$, then $I_{(\sigma; \rho)/\chi}(\forall v . F) = \text{unknown}$. (The verification of this is an easy exercise.) But since this result has very limited applicability (only when there are two system objects), it was not worth stating as a lemma. Conversely, the same type of situation produced by the aforementioned counterexample would obtain for conjunctions if they were allowed to have arbitrarily many components, e.g., as in $F = F_1 \land F_2 \land F_3$. Then if $I_{(\sigma; \rho)/\chi}(F_1) = \text{true}$ but $I_{(\sigma; \rho)/\chi}(F_2) = I_{(\sigma; \rho)/\chi}(F_3) = \text{unknown}$, we could well have $I_{(\sigma; \rho)/\chi}(F) = \text{false}$, because the two unknowns ($F_2$ and $F_3$) might result in false, which will then of course falsify the entire conjunction. Loosely speaking, every time two or more unknowns are combined conjunctively, whether in a conjunction proper or in a universal generalization, the result might be false.

The following is the analogue of the preceding lemma for existential generalizations, corresponding to Lemma 12.

**Lemma 14:** (a) $I_{(\sigma; \rho)/\chi}(\exists v . F) = \text{false}$ iff $I_{(\sigma; \rho)/\chi[v\mapsto s_i]}(F) = \text{false}$ for every $i = 1, \ldots, n$.
(b) If $I_{(\sigma; \rho)/\chi[v\mapsto s_i]}(F) = \text{true}$ for some $i \in \{1, \ldots, n\}$, then

$$I_{(\sigma; \rho)/\chi}(\exists v . F) = \text{true}$$

However, the converse is not true. In particular, we may have

$$I_{(\sigma; \rho)/\chi}(\exists v . F) = \text{true}$$

even though

$$I_{(\sigma; \rho)/\chi[v\mapsto s_i]}(F) \neq \text{true}$$

for all $i \in \{1, \ldots, n\}$.

**Proof:** For part (a):

$$I_{(\sigma; \rho)/\chi}(\exists v . F) = \text{false} \iff (1.36)$$

$$\forall \ w \subseteq \sigma . \ V_{(\sigma; \rho)/\chi}^I(\exists v . F) = \text{false} \iff (1.35)$$

$$\forall \ w \subseteq \sigma . \ \forall i \in \{1, \ldots, n\} . \ V_{(\sigma; \rho)/\chi[v\mapsto s_i]}^I[F] = \text{false} \iff \text{(quantifier swap)}$$

$$\forall i \in \{1, \ldots, n\} . \ \forall \ w \subseteq \sigma . \ V_{(\sigma; \rho)/\chi[v\mapsto s_i]}^I[F] = \text{false} \iff (1.36)$$

For (b), assume

$$I_{(\sigma; \rho)/\chi[v\mapsto s_i]}(F) = \text{true}.$$ 

Then, by (1.36),

$$\forall \ w \subseteq \sigma . \ V_{(\sigma; \rho)/\chi[v\mapsto s_i]}^I[F] = \text{true},$$

thus

$$\forall \ w \subseteq \sigma . \ V_{(\sigma; \rho)/\chi}^I(\exists v . F) = \text{true},$$

i.e., $I_{(\sigma; \rho)/\chi}(\exists v . F) = \text{true}$. To see that the converse does not hold, consider again the system state $\sigma$ specified in the proof of Lemma 13.

---

Part (d) of Lemma 11 has a straightforward analogue in the case of Lemma 13 which was not stated explicitly in the latter because it is rather trivial.
Thus $\sigma$ contains the following two worlds, $w_1$ and $w_2$:

\[
\begin{array}{c|cc}
\sigma: & G & R \\
\hline
s_1 & \{G, R\} \\
R & G \\
\hline
s_2 & s_3
\end{array}
\]

\[
\begin{align*}
w_1(\text{color}, s_1) &= G & w_2(\text{color}, s_1) &= R \\
w_1(\text{color}, s_2) &= R & w_2(\text{color}, s_2) &= R \\
w_1(\text{color}, s_3) &= G & w_2(\text{color}, s_3) &= G
\end{align*}
\]

Now let $\rho$ and $\chi$ be as described before (i.e., $\rho(r_1) = s_1$ and $\chi$ arbitrary), and let $F$ be the formula

\[
\exists u . \text{adjacent}(u, r_1) \land u \neq r_1 \land \neg \text{sameColor}(u, r_1).
\]

Let $M$ be the body of the above formula, i.e.,

\[
M = \text{adjacent}(u, r_1) \land u \neq r_1 \land \neg \text{sameColor}(u, r_1).
\]

The reader will verify that

\[
I_{(\sigma; \rho)/\chi}(\exists u . M) = \text{true},
\]

because

\[
\mathcal{V}^I_{(w_1; \rho)/\chi}[F] = \text{true} \quad \text{and} \quad \mathcal{V}^I_{(w_2; \rho)/\chi}[F] = \text{true}.
\]

However:

\[
\begin{align*}
I_{(\sigma; \rho)/\chi}[u \mapsto s_1](M) & = \text{false}; \\
I_{(\sigma; \rho)/\chi}[u \mapsto s_2](M) & = \text{unknown}; \\
I_{(\sigma; \rho)/\chi}[u \mapsto s_3](M) & = \text{unknown}.
\end{align*}
\]

Briefly, (1.72) holds because $u$ and $r_1$ are coreferential under $\rho$ and $\chi[u \mapsto s_1]$, and thus $u \neq r_1$ is false; (1.73) holds because

\[
\mathcal{V}^I_{(w_1; \rho)/\chi}[M] = \text{true} \quad \text{while} \quad \mathcal{V}^I_{(w_2; \rho)/\chi}[M] = \text{false};
\]

and (1.74) holds because

\[
\mathcal{V}^I_{(w_1; \rho)/\chi}[M] = \text{false} \quad \text{while} \quad \mathcal{V}^I_{(w_2; \rho)/\chi}[M] = \text{true}.
\]

This establishes that the converse of (b) does not obtain.}

For any given $F$, $\rho$, and $\chi$, the basis of $F$ w.r.t $\rho$ and $\chi$, denoted $B(F, \rho, \chi)$, is a set of a.o. pairs (or an error token $\infty$). It is defined by structural recursion on $F$, as shown below. The first clause covers atomic formulas, for an arbitrary relation symbol of arity $n$ and profile $\text{Prof}(R) = [(l_{i_1}; j_1) \cdot \cdots (l_{i_m}; j_m)]$:

\[
\begin{align*}
B(\text{true}, \rho, \chi) & = \emptyset; \\
B(\text{false}, \rho, \chi) & = \emptyset; \\
B(R(t_1, \ldots, t_n), \rho, \chi) & = \left\{ \begin{array}{ll}
\{(l_{i_1}; t_{j_1}^\rho), \ldots, (l_{i_m}; t_{j_m}^\rho)\} & \text{if } t_{j_k}^\rho \downarrow \text{ for every } k \in \{1, \ldots, m\}; \\
\infty & \text{otherwise.}
\end{array} \right.
\end{align*}
\]

\[
\begin{align*}
B(\neg F, \rho, \chi) & = B(F, \rho, \chi); \\
B(F_1 \land F_2, \rho, \chi) & = B(F_1, \rho, \chi) \cup B(F_2, \rho, \chi); \\
B(F_1 \lor F_2, \rho, \chi) & = B(F_1, \rho, \chi) \cup B(F_2, \rho, \chi); \\
B(\forall v . F, \rho, \chi) & = \bigcup_{i=1}^{n} B(F, \rho, \chi[v \mapsto s_i]); \\
B(\exists v . F, \rho, \chi) & = \bigcup_{i=1}^{n} B(F, \rho, \chi[v \mapsto s_i]).
\end{align*}
\]

32
To keep the notation simple, we have assumed that the set-theoretic union operation in the above identities is strict w.r.t. to $\infty$, i.e., that $S_1 \cup \cdots \cup S_n = \infty$ whenever $S_i = \infty$ for some $i \in \{1, \ldots, n\}$.

For any two lists $L_1$ and $L_2$, and any set $S$ of positive integers, define $L_1 \equiv_S L_2$ as follows:

$$L_1 \equiv_S L_2 \iff [\forall x \in S . L_1(x) = L_2(x)].$$  \hfill (1.75)

(Nota that $L_1 \equiv_S L_2$ for any lists $L_1$ and $L_2.$) That is, two lists $L_1$ and $L_2$ are identical w.r.t. a set of positions $S$ iff they have identical elements in each position in $S$. It is easily verified that this is an equivalence relation (for fixed $S$). Now, likewise, for any two worlds $w_1$ and $w_2$, and any set $S$ of a.o. pairs, define

$$w_1 \equiv_S w_2 \iff [\forall (l; s) \in S . w_1(l, s) = w_2(l, s)].$$  \hfill (1.76)

If we think of worlds as strings, then the above definition is isomorphic to (1.75), with a.o. pairs playing the role of string positions. The following is a trivial consequence of the above definition, but useful enough to isolate as a lemma:

**Lemma 15:** If $w_1 \equiv_S w_2$ then $w_1 \equiv_{S'} w_2$ for every $S' \subseteq S$.

The next result has important efficiency implications for formula evaluation. Its practical upshot is that in evaluating a formula $F$ is a certain state $\sigma$ and assignments $\rho$ and $\chi$, we only need to consider worlds that differ along the coordinates in $B(F, \rho, \chi)$. For any $w \subseteq \sigma$, attribute values of $w$ for a.o. pairs not in $B(F, \rho, \chi)$ can be ignored. As will be illustrated, the resulting savings can be significant.

**Theorem 16:** If $B(F, \rho, \chi) \neq \infty$ and $w_1 \equiv B(F, \rho, \chi) w_2$, then $V^I(w_1; \rho/\chi)[F] = V^I(w_2; \rho/\chi)[F]$.

**Proof:** By induction on the structure of $F$. Suppose first that $F$ is an atomic formula $R(t_1, \ldots, t_n)$, where

$$\text{Prof}(R) = [(l_{i_1}; j_1) \cdots (l_{i_m}; j_m)].$$

We first note that $B(F, \rho, \chi) \neq \infty$ implies $t^\rho_{j_k} \downarrow$ for every $k = 1, \ldots, m$. Now, the hypothesis

$$w_1 \equiv B(F, \rho, \chi) w_2$$

entails

$$w_1(l_{i_1}, t^\rho_{j_1}) = w_2(l_{i_1}, t^\rho_{j_1}), \ldots, w_1(l_{i_m}, t^\rho_{j_m}) = w_2(l_{i_m}, t^\rho_{j_m}),$$

hence

$$R^I(w_1(l_{i_1}, t^\rho_{j_1}), \ldots, w_1(l_{i_m}, t^\rho_{j_m})) \iff R^I(w_2(l_{i_1}, t^\rho_{j_1}), \ldots, w_2(l_{i_m}, t^\rho_{j_m})).$$

Since $t^\rho_{j_k} \downarrow$ for all $k \in \{1, \ldots, m\}$, (1.30) entails

$$V^I(w_1; \rho/\chi)[R(t_1, \ldots, t_n)] = V^I(w_2; \rho/\chi)[R(t_1, \ldots, t_n)].$$

Next, suppose that $F$ is a negation of the form $\neg G$. We have $B(F, \rho, \chi) = B(G, \rho, \chi)$, hence

$$w_1 \equiv B(G, \rho, \chi) w_2,$$

and therefore, inductively,

$$V^I(w_1; \rho/\chi)[G] = V^I(w_2; \rho/\chi)[G].$$

Hence, by (1.31),

$$V^I(w_1; \rho/\chi)[\neg G = F] = V^I(w_2; \rho/\chi)[\neg G = F].$$

When $F$ is a conjunction of the form $F_1 \land F_2$, we have $B(F_i, \rho, \chi) \subseteq B(F, \rho, \chi)$ for $i \in \{1, 2\}$, so Lemma 15 gives

$$w_1 \equiv B(F_i, \rho, \chi) w_2.$$

Hence, since $B(F_i, \rho, \chi) \neq \infty$ \footnote{Otherwise we would have $B(F, \rho, \chi) = \infty.$}, the inductive hypothesis yields

$$V^I(w_1; \rho/\chi)[F_1] = V^I(w_2; \rho/\chi)[F_1]$$

and

$$V^I(w_1; \rho/\chi)[F_2] = V^I(w_2; \rho/\chi)[F_2].$$
Further, let which, by (1.34), implies
\[ \forall s_i \in \{1, \ldots, n\} \cdot B(G, \rho, \chi[v \mapsto s_i]) \subseteq B(F, \rho, \chi). \]
Lemma 13 implies
\[ \forall i \in \{1, \ldots, n\} \cdot w_1 = B(G, \rho, \chi[v \mapsto s_i], w_2. \]
Further, \( B(G, \rho, \chi[v \mapsto s_i]) \neq \infty \) for \( i = 1, \ldots, n \). Hence, the inductive hypothesis yields:
\[ \forall i \in \{1, \ldots, n\} \cdot V_{(w_1; \rho, \chi[v \mapsto s_i])}[G] = V_{(w_2; \rho, \chi[v \mapsto s_i])}[G], \]
which, by (1.34), implies \( V_{(w_1; \rho, \chi)}[\forall v . G] = V_{(w_2; \rho, \chi)}[\forall v . G] \). Existentially quantified formulas are handled in a similar manner.

To illustrate the utility of the above result, consider again the system of the two clocks \( s_1 \) and \( s_2 \) of Example 12, and let \( \sigma \) be the following state of that system:
\[
\begin{align*}
\sigma(\text{hours, } s_1) &= \{9, 16, 17\}; \\
\sigma(\text{minutes, } s_1) &= \{0, 1, 2, \ldots, 59\}; \\
\sigma(\text{hours, } s_2) &= \{5, 18\}; \\
\sigma(\text{minutes, } s_2) &= \{10, 11, \ldots, 59\}.
\end{align*}
\]

Viewed as an ADFA, \( \sigma \) can be depicted as follows:

This state contains \( 3 \cdot 60 \cdot 2 \cdot 50 = 18,000 \) worlds. Consider now the formula \( F = \forall x . \ PM(x) \), asserting that every clock’s time is p.m. This formula is true in some worlds (e.g., when \( \text{hours}(s_1) = 17, \text{minutes}(s_1) = 2, \text{hours}(s_2) = 18, \text{minutes}(s_2) = 39 \)), and false in others (e.g., when \( \text{hours}(s_1) = 9, \text{minutes}(s_1) = 44, \text{hours}(s_2) = 18, \text{minutes}(s_2) = 4 \)).

Thus, as is readily verified, we have:
\[ I_{(\sigma; \rho, \chi)}[\forall x . \ PM(x)] = \text{unknown} \] (1.77)
for arbitrary \( \rho \) and \( \chi \). However, if we simply tried to compute \( I_{\rho/\chi[s \mapsto s_i]}(\PM(x)) \) and \( I_{\rho/\chi[s \mapsto s_j]}(\PM(x)) \), we would obtain \text{unknown} and \text{unknown}, respectively, and therefore, by part (b) of Lemma 13 we would not be able to infer (1.77), because it would not be possible to rule out the possibility \( I_{(\sigma; \rho, \chi)}[\forall x . \ PM(x)] = \text{false} \). Accordingly, we would need to resort to calculating
\[ V_{(w_1; \rho, \chi)}[\forall x . \ PM(x)] \]
for every \( w \subseteq \sigma \). But this could be highly inefficient. Depending on how we iterate through the space of 18,000 worlds, we might have to evaluate \( \forall x . \ PM(x) \) over one hundred times before we find two worlds \( w_i \) and \( w_j \) such that
\[ V_{(w_i; \rho, \chi)}[\forall x . \ PM(x)] = \text{true} \quad \text{and} \quad V_{(w_j; \rho, \chi)}[\forall x . \ PM(x)] = \text{false}, \]
which would allow us to arrive at the answer (1.77).

But examining all these worlds is unnecessary for a simple reason: The \text{minutes} attribute of a clock is irrelevant in determining whether or not the clock’s time is p.m. Only the value of \text{hours} is necessary for making that judgment. If that value is greater than 11, then the clock is p.m., otherwise it is not. By taking advantage of this information, we can restrict attention only to the \text{hours} attribute of the two clocks. This is precisely the information provided by the basis of \( F \):
Arbitrary values can be selected for the minutes attribute, say 0 for minutes(s₁) and 10 for minutes(s₂). We can then narrow down the relevant space from 18,000 worlds to only 6 worlds w₁,...,w₆, formed by choosing a value for hours(s₁) (three choices) followed by a value for hours(s₂) (two choices), and keeping minutes(s₁) and minutes(s₂) fixed to 0 and 10, respectively:

\[
\begin{align*}
  w₁(\text{hours, s₁}) & = 9 & w₁(\text{minutes, s₁}) & = 0 & w₁(\text{hours, s₂}) & = 5 & w₁(\text{minutes, s₂}) & = 10; \\
  w₂(\text{hours, s₁}) & = 9 & w₂(\text{minutes, s₁}) & = 0 & w₂(\text{hours, s₂}) & = 18 & w₂(\text{minutes, s₂}) & = 10; \\
  w₃(\text{hours, s₁}) & = 16 & w₃(\text{minutes, s₁}) & = 0 & w₃(\text{hours, s₂}) & = 5 & w₃(\text{minutes, s₂}) & = 10; \\
  w₄(\text{hours, s₁}) & = 16 & w₄(\text{minutes, s₁}) & = 0 & w₄(\text{hours, s₂}) & = 18 & w₄(\text{minutes, s₂}) & = 10; \\
  w₅(\text{hours, s₁}) & = 17 & w₅(\text{minutes, s₁}) & = 0 & w₅(\text{hours, s₂}) & = 5 & w₅(\text{minutes, s₂}) & = 10; \\
  w₆(\text{hours, s₁}) & = 17 & w₆(\text{minutes, s₁}) & = 0 & w₆(\text{hours, s₂}) & = 18 & w₆(\text{minutes, s₂}) & = 10.
\end{align*}
\]

Now \(\forall \; x \cdot \text{PM}(x)\) will only need to be evaluated in four worlds \((w₁—w₄)\) before we determine the correct answer, \((1.77)\).

Computing \(\mathcal{B}(F, \rho, \chi)\) allows us to perform this type of narrowing in a systematic and sound way. Roughly, atomic formulas \(R(t₁,...,tₙ)\) are restricted to appropriate a.o. pairs by utilizing the profile information of R, while the basis of a complex formula is computed recursively (and conservatively) by joining the bases of each subformula. Since computing the basis of a formula is very cheap (linear both in time and space), we can “statically analyze” every formula \(F\) that needs to be evaluated in a given state \(\sigma\) and assignments \(\rho\) and \(\chi\) by computing \(\mathcal{B}(F, \rho, \chi)\), picking arbitrary values for the a.o. pairs which are not in the basis, and constructing all and only those worlds \(w'\) which reflect different combinations of values for a.o. pairs in the basis. Theorem \((1.6)\) then allows us to evaluate \(F\) only w.r.t. to such worlds \(w'\).

**Definition 10:** A world \((w; \hat{\rho})\) satisfies a formula \(F\) w.r.t. a variable assignment \(\chi\) iff

\[ \mathcal{V}_{(w; \hat{\rho})/\chi}[F] = \text{true}. \]

We denote this by writing \((w; \hat{\rho}) \models_{\chi} F\). Likewise, we say that a world \((w; \hat{\rho})\) satisfies a named state \((\sigma; \rho)\) iff \((w; \hat{\rho}) \subseteq (\sigma; \rho)\). This is denoted by \((w; \hat{\rho}) \models (\sigma; \rho)\). We say that \((w; \hat{\rho})\) satisfies a context \(\gamma = (\beta; (\sigma; \rho))\) w.r.t. a given \(\chi\), written \((w; \hat{\rho}) \models_{\chi} \gamma\), iff \((w; \hat{\rho}) \models_{\chi} F\) for every \(F \in \beta\) and \((w; \hat{\rho}) \models (\sigma; \rho)\). Finally, we say that a context \(\gamma\) entails a formula \(F\), written \(\gamma \models F\), iff \((w; \hat{\rho}) \models_{\chi} \gamma\) implies \((w; \hat{\rho}) \models_{\chi} F\) for all worlds \((w; \hat{\rho})\) and variable assignments \(\chi\). Likewise, \(\gamma\) entails a named state \((\sigma' ; \rho')\), written \(\gamma \models (\sigma' ; \rho')\), iff, for all worlds \((w; \hat{\rho})\) and variable assignments \(\chi\), we have \((w; \hat{\rho}) \models (\sigma' ; \rho')\) whenever \((w; \hat{\rho}) \models_{\chi} \gamma\).

**Lemma 17 (Weakening):** If \(\beta; (\sigma; \rho)\) \models F then \((\beta \cup \beta'; (\sigma; \rho)) \models F\); and if \((\beta; (\sigma; \rho)) \models (\sigma' ; \rho')\) then

\[ \mathcal{B}(\beta; (\sigma; \rho)) \models (\sigma' ; \rho'). \]

**Lemma 18:** If \((\beta; (\sigma; \rho)) \models (\sigma' ; \rho')\) and \((\beta; (\sigma'; \rho')) \models F\) then \((\beta; (\sigma; \rho)) \models F\).

**Proof:** Pick any world \((w; \hat{\rho})\) and variable assignment \(\chi\) and suppose that

\[ (w; \hat{\rho}) \models_{\chi} (\beta; (\sigma; \rho)). \] (1.78)
Then, by the assumption \((\beta; (\sigma; \rho)) \models (\sigma'; \rho')\), we conclude
\[
(w; \tilde{\rho}) \models (\sigma'; \rho').
\] (1.79)

From (1.78) and (1.79) we infer
\[
(w; \tilde{\rho}) \models \chi(\beta; (\sigma'; \rho')).
\] (1.80)

Finally, (1.80) and the assumption \((\beta; (\sigma'; \rho')) \models F\) imply \((w; \tilde{\rho}) \models \chi F\).

**Lemma 19:** \((\beta; (\sigma; \rho)) \models (\sigma; \rho)\).

**Lemma 20:** \(((\beta \cup \{false\}); (\sigma; \rho)) \models (\sigma'; \rho')\).

**Proof:** Pick any world \((w; \tilde{\rho})\) and variable assignment \(\chi\), and assume
\[
(w; \tilde{\rho}) \models \chi(\beta \cup \{false\}; (\sigma; \rho)),
\]
so that \((w; \tilde{\rho}) \models \chi false\), i.e., \(\mathcal{V}^I_{(w; \tilde{\rho})/\chi}[false] = true\). But, by definition (1.29), \(\mathcal{V}^I_{(w; \tilde{\rho})/\chi}[false] = false\), and the contradiction entitles us to infer \((w; \tilde{\rho}) \models (\sigma'; \rho')\).

**Lemma 21:** If \((\beta; (\sigma; \rho)) \models (\sigma'; \rho')\) and \((\sigma'; \rho') \subseteq (\sigma''; \rho'')\) then \((\beta; (\sigma; \rho)) \models (\sigma''; \rho'')\).

**Proof:** Pick any world \((w; \tilde{\rho})\) and variable assignment \(\chi\) and suppose that
\[
(w; \tilde{\rho}) \models \chi(\beta; (\sigma; \rho)).
\] (1.81)

From the assumption \((\beta; (\sigma; \rho)) \models (\sigma'; \rho')\) and (1.81) we obtain \((w; \tilde{\rho}) \models (\sigma'; \rho')\), which is to say \((w; \tilde{\rho}) \subseteq (\sigma'; \rho')\).

Finally, \((w; \tilde{\rho}) \subseteq (\sigma'; \rho')\), the assumption \((\sigma'; \rho') \subseteq (\sigma''; \rho'')\) and Lemma 20 yield
\[
(w; \tilde{\rho}) \subseteq (\sigma''; \rho''),
\]
i.e., \((w; \tilde{\rho}) \models (\sigma''; \rho'')\).

**Corollary 22** (Widening is sound): If \((\sigma; \rho) \subseteq (\sigma'; \rho')\) then \((\beta; (\sigma; \rho)) \models (\sigma'; \rho')\).

**Proof:** By Lemma 19 \((\beta; (\sigma; \rho)) \models (\sigma; \rho)\), hence, by Lemma 21 \((\beta; (\sigma; \rho)) \models (\sigma'; \rho')\).

We now extend the notion of alternative extensions that was introduced in Section 1.3 to named states.

**Definition 11:** Let \((\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m), (\sigma'; \rho') \in (\sigma; \rho), m \geq 1\). We say that \((\sigma'; \rho')\) is an alternative extension of \((\sigma; \rho)\) w.r.t. \((\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\), written
\[
Alt((\sigma; \rho), \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}, (\sigma'; \rho')),
\]
iff \(Dom(\rho') = Dom(\rho_1) \cup \cdots \cup Dom(\rho_m)\) and there is a subset \(S \subseteq \{1, \ldots, m\}\) such that:

1. \(\rho'\) conflicts with \(\rho_i\) if \(i \in S\); and
2. if \(S \neq \{1, \ldots, m\}\) then \(Alt(\sigma, \{\sigma_i \mid i \in \{1, \ldots, m\} \setminus S\}, \sigma')\), while if \(S = \{1, \ldots, m\}\) then \(\sigma' = \sigma\).

Owing to the first condition, if such a subset \(S \subseteq \{1, \ldots, m\}\) exists, then it is unique. When \(m = 1\) we might write \(Alt((\sigma; \rho), (\sigma_1; \rho_1), (\sigma'; \rho'))\) instead of \(Alt((\sigma; \rho), \{(\sigma_1; \rho_1)\}, (\sigma'; \rho'))\).

The following algorithm computes all alternative extensions of \((\sigma; \rho)\) w.r.t. \((\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\):

1. Let \(\rho_1^a, \ldots, \rho_k^a, k \geq 1\), be all and only the constant assignments on \(Dom(\rho_1) \cup \cdots \cup Dom(\rho_m)\) that are supersets of \(\rho\). Note that there are \(k = n^d\) such assignments, where \(n\) is the number of system objects and
\[
d = |[Dom(\rho_1) \cup \cdots \cup Dom(\rho_m)] \setminus Dom(\rho)|.
\]
2. Let \(R = \emptyset\).
Lemma 23: \( \text{for if } \rho \text{ and let } \Phi_i = \{ \sigma \}, \text{ else } \Phi_i = \text{AE}(\Sigma_i, \sigma). \)

Set \( R \leftarrow R \cup \{(\sigma'; \rho'_i) \mid \sigma' \in \Phi_i \}. \)

4. Return \( R. \)

The algorithm is rather naive in that it may duplicate some work in the process of computing \( \text{AE}(\Sigma_i, \sigma) \) for the various \( i. \) In practice the values of \( \text{AE}(\Sigma_i, \sigma) \) would be memorized.

**Lemma 23:** \( (\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m) \subseteq (\sigma; \rho), m \geq 1, (w; \tilde{\rho}) \subseteq (\sigma; \rho), \text{ and } \)

\[ \forall i \in \{1, \ldots, m\} . (w; \tilde{\rho}) \not\subseteq (\sigma_i; \rho_i) , \]

then there is \( (\sigma'; \rho') \supseteq (\sigma; \rho) \) such that \( \text{Alt}((\sigma; \rho), \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}, (\sigma'; \rho')) \) and \( (w; \tilde{\rho}) \subseteq (\sigma'; \rho'). \)

**Proof:** The following holds by assumption:

\[ \forall i \in \{1, \ldots, m\} . w \not\subseteq \sigma_i \lor \tilde{\rho} \not\supseteq \rho_i , \]

Define

\[ S = \{ i \in \{1, \ldots, m\} \mid \tilde{\rho} \not\supseteq \rho_i \} \]

and let

\[ \rho' = \tilde{\rho} \restriction [\text{Dom}(\rho_1) \cup \cdots \cup \text{Dom}(\rho_m)], \]

so that

\[ \tilde{\rho} \supseteq \rho' \supseteq \rho. \]

It follows by construction that

\[ \text{Dom}(\rho') = \text{Dom}(\rho_1) \cup \cdots \cup \text{Dom}(\rho_m) \]

and

\[ \forall i \in \{1, \ldots, m\} . \rho' \not\supseteq \rho_i \iff i \in S , \]

which is to say that \( \rho' \) has a conflict with \( \rho_i \) iff \( i \in S. \)

There are two cases to consider here: \( S = \{1, \ldots, m\} \) or \( S \subset \{1, \ldots, m\}. \) In the first case, we must have

\[ \rho' \supseteq (\rho, \]

for if \( \rho' = \rho \) then, from \([1.84]\), \( \rho_1 = \cdots = \rho_m = \rho \) and hence \( \tilde{\rho} \supseteq \rho_i \) for all \( i = 1, \ldots, m, \) since \( \rho \subseteq \tilde{\rho} \) by assumption.

But, from \([1.83]\), \( \forall i \in \{1, \ldots, m\} . \tilde{\rho} \supseteq \rho_i \) would entail \( S = \emptyset, \) contradicting the supposition \( S = \{1, \ldots, m\} \) (recall that \( m \geq 1). \) Define \( \sigma' = \sigma. \) Then \( (\sigma'; \rho') \subseteq (\sigma; \rho) \) by \([1.85]\) and \( \sigma \subseteq \sigma, \) and indeed \( (\sigma'; \rho') \supseteq (\sigma; \rho) \) by \([1.86]\) and \([1.84]\). By construction, \( \text{Alt}((\sigma; \rho), \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}, (\sigma'; \rho')) \) and \( (w; \tilde{\rho}) \subseteq (\sigma'; \rho'). \) In addition, \( (w; \tilde{\rho}) \subseteq (\sigma'; \rho') \) follows from \( w \subseteq \sigma \) and \([1.85].\)

By contrast, suppose that \( S \subset \{1, \ldots, m\}, \) so that

\[ \{1, \ldots, m\} \setminus S \neq \emptyset . \]

From the definition of \( S \) and \([1.82]\) we obtain

\[ \forall i \in \{1, \ldots, m\} \setminus S . w \not\subseteq \sigma_i. \]

From \([1.88]\) we infer that

\[ \forall i \in \{1, \ldots, m\} \setminus S . \sigma_i \subseteq \sigma , \]

\[ \geq 4. \]
for otherwise there would be some \( j \in \{1, \ldots, m\} \setminus S \) such that \( \sigma_j = \sigma \). But \( w \subsetneq \sigma = \sigma_j \) contradicts the assumption \( w \not\subseteq \sigma_j \). Further, we must have \( w \subsetneq \sigma \), for, in light of (1.87), \( w = \sigma \) would contradict (1.89), given that worlds do not have any proper extensions. Therefore, by Lemma 5, there exists a state

\[ \sigma' \subseteq \sigma \]  

(1.90)
such that \( Alt(\sigma, \{\sigma_i \mid i \in \{1, \ldots, m\} \setminus S\}, \sigma') \) and

\[ w \subseteq \sigma'. \]  

(1.91)
From (1.90), (1.84), and (1.85), we conclude \((\sigma'; \rho') \supseteq (\sigma; \rho)\). Moreover, by construction,

\[ Alt((\sigma; \rho), \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}, (\sigma'; \rho')) \]

while \((w; \hat{\rho}) \subseteq (\sigma'; \rho')\) follows from (1.91) and (1.85). This concludes the case analysis.

**Corollary 24:** If \((\sigma'; \rho') \supseteq (\sigma; \rho), (w; \hat{\rho}) \subseteq (\sigma; \rho)\), and \((w; \hat{\rho}) \not\subseteq (\sigma'; \rho')\) then there is some

\((\sigma''; \rho'') \supseteq (\sigma; \rho)\)
such that \( Alt((\sigma; \rho), (\sigma'; \rho'), (\sigma''; \rho''))\) and \((w; \hat{\rho}) \subseteq (\sigma''; \rho'')\).

We end this section by introducing the following notion of state entailment:

**Definition 12:** Suppose that \((\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m) \subseteq (\sigma; \rho)\) and let \( \beta \) be any assumption base. We say that \((\sigma; \rho)\) **entails** \((\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\) w.r.t. \( \beta \), written \((\sigma; \rho) \vDash_{\beta} \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}\), iff for every \((\sigma'; \rho')\) such that

\[ Alt((\sigma; \rho), \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}, (\sigma'; \rho')) \]

the following holds for all \( \chi \):

\[ I_{(\sigma'; \rho')/\chi} \left( \bigwedge_{F \in \beta} F \right) \text{ false.} \]

When \( n = 1 \) we drop the braces and write \((\sigma; \rho) \vDash_{\beta} (\sigma_1; \rho_1)\) instead of \((\sigma; \rho) \vDash_{\beta} \{(\sigma_1; \rho_1)\}\).

This definition captures the intuition that any world which extends the state \((\sigma; \rho)\) and satisfies the formulas in \( \beta \) must also extend one of the states \((\sigma_i; \rho_i)\), in the sense that any alternative way of extending \((\sigma; \rho)\) will end up falsifying some element of \( \beta \). (Of course, if there are no alternative ways of extending \((\sigma; \rho)\) then the entailment holds vacuously, even if \( \beta = \emptyset \).) This is formally demonstrated by the proof of Lemma 25 below.

Determining whether or not \((\sigma; \rho) \vDash_{\beta} \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}\) is decidable; we present an algorithm for it which makes use of an auxiliary function \( g \) that takes a formula \( F \) and a named state \((\sigma; \rho)\) and returns true or false. To compute \( g(F, (\sigma; \rho)) \):

1. Let \( \psi_1, \ldots, \psi_k \) be all distinct functions from \( FV(F) \) to the set of system objects \( \{s_1, \ldots, s_n\} \). (There are \( k = n|FV(F)| \) such functions.)

2. Let \( \chi_1, \ldots, \chi_k \) be arbitrary variable assignments such that

\[ \forall i \in \{1, \ldots, k\} \cdot \chi_i \upharpoonright FV(F) = \psi_i. \]

3. If \( I_{(\sigma; \rho)/\chi}(F) = \text{false} \) for every \( i = 1, \ldots, k \) then return true, else return false.

The algorithm for determining \((\sigma; \rho) \vDash_{\beta} \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}\) can now be stated thus:

1. For each \((\sigma'; \rho')\) such that \( Alt((\sigma; \rho), \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}, (\sigma'; \rho')) \):
   - If
     \[ g \left( \bigwedge_{F \in \beta} F, (\sigma'; \rho') \right), \]
     then continue, else return false.
2. Return true.

The algorithm clearly hinges on $g$, whose correctness in this context depends on Lemma 7.

**Lemma 25:** If $(\sigma; \rho) \models_{\beta} \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}$ then for all worlds $(w; \widehat{\rho})$ and variable assignments $\chi$, if $(w; \widehat{\rho}) \models_{\chi} (\beta; (\sigma; \rho))$

there is some $i \in \{1, \ldots, m\}$ such that $(w; \widehat{\rho}) \models (\sigma_i; \rho_i)$.

**Proof:** Assuming $(\sigma; \rho) \models_{\beta} \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}$, pick any world $(w; \widehat{\rho})$ and variable assignment $\chi$ and suppose that $(w; \widehat{\rho}) \models_{\chi} (\beta; (\sigma; \rho))$ so that

$$(w; \widehat{\rho}) \subseteq (\sigma; \rho)$$

and

$$\forall F \in \beta . \mathcal{V}_{(w; \widehat{\rho})/\chi}^I[F] = \text{true}. \quad (1.92)$$

It follows that

$$\mathcal{V}_{(w; \widehat{\rho})/\chi}^I \left[ \bigwedge_{F \in \beta} F \right] = \text{true}. \quad (1.93)$$

By way of contradiction, suppose that there is no $i \in \{1, \ldots, m\}$ such that $(w; \widehat{\rho}) \models (\sigma_i; \rho_i)$, i.e.,

$$\forall i \in \{1, \ldots, m\} . (w; \widehat{\rho}) \not\subseteq (\sigma_i; \rho_i).$$

By Lemma 23 there is some

$$(\sigma'; \rho') \cong (\sigma; \rho)$$

such that

$$\text{Alt}((\sigma; \rho), \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}, (\sigma'; \rho'))$$

and

$$(w; \widehat{\rho}) \subseteq (\sigma'; \rho'). \quad (1.94)$$

But then, by Definition 12 and (1.96) it follows that

$$I_{(\sigma'; \rho')/\chi} \left[ \bigwedge_{F \in \beta} F \right] = \text{false}. \quad (1.95)$$

Hence, from (1.36) and $w \subseteq \sigma'$, we get

$$\mathcal{V}_{(w; \rho')/\chi}^I \left[ \bigwedge_{F \in \beta} F \right] = \text{false},$$

and from $\widehat{\rho} \supseteq \rho'$ and Lemma 5

$$\mathcal{V}_{(w; \widehat{\rho})/\chi}^I \left[ \bigwedge_{F \in \beta} F \right] = \text{false},$$

which contradicts (1.95).

**Corollary 26:** If $(\sigma; \rho) \models_{\beta} (\sigma'; \rho')$ then $(\beta; (\sigma; \rho)) \models (\sigma'; \rho').
1.6 A family of diagrammatic natural deduction languages

We now formally define Vivid, a family of natural deduction languages in the DPL tradition (Arkoudas 2000) that combine sentential and diagrammatic reasoning. A concrete instance of Vivid is obtained by specifying a vocabulary \( \Sigma = (C,R,V) \), an attribute structure \( A = (\{l_1 : A_1, \ldots, l_k : A_k\}; R) \), and an interpretation \( I \) of \( R \) into \( A \). We assume in what follows that \( \Sigma, A, \) and \( I \) have been fixed. The terms and formulas of the language are as described in Section 1.5.

We write \( F[t/v] \) to denote the formula obtained from \( F \) by replacing every free occurrence of \( v \) by the term \( t \) (taking care to rename \( F \) if necessary to avoid variable capture). The following result is readily proved by induction on the structure of \( F \).

\[ \text{Lemma 27: If } b \in \{\text{true, false}\}, \quad \forall^I_{(u; \rho)/\chi[v \mapsto s]}[F] = b, \]

and \( v' \) does not occur in \( F \), then

\[ \forall^I_{(u; \rho)/\chi[v' \mapsto s]}[F[v'/v]] = b. \]

1.6.1 Abstract syntax

There are two syntactic categories of proofs, sentential and diagrammatic. Sentential deductions are used to derive formulas, while diagrammatic deductions are used to derive diagrams. We will see that the two can be freely mixed, and indeed that their structures are mutually recursive. We use the letters \( D \) and \( \Delta \) to range over sentential and diagrammatic deductions, respectively. The symbol \( \mathcal{D} \) will range over the union of the two. The abstract syntax (Reynolds 1998) of both proof types is defined by the grammars below:

\[
D ::= \text{RuleApp} \\
\quad | \text{assume } F \ D \\
\quad | \text{claim } F \\
\quad | \text{true-intro} \\
\quad | \text{false-elim} \\
\quad | \text{modus-ponens } F \Rightarrow G, F \\
\quad | \text{modus-tollens } F \Rightarrow G, \neg G \\
\quad | \text{double-negation } \neg \neg F \\
\quad | \text{absurd } F, \neg F \\
\quad | \text{left-and } F \land G \\
\Delta ::= \mathcal{D}; \Delta \\
\quad | \text{claim } (\sigma; \rho) \\
\quad | (\sigma; \rho) \text{ by thinning with } F_1, \ldots, F_n \\
\quad | (\sigma; \rho) \text{ by widening} \\
\quad | (\sigma; \rho) \text{ by absurdity} \\
\quad | \text{cases } F_1, \ldots, F_k: (\sigma_1; \rho_1) \rightarrow D_1 | \cdots | (\sigma_n; \rho_n) \rightarrow D_n \\
\quad | \text{cases } F_1 \lor F_2: F_1 \rightarrow D_1 | F_2 \rightarrow D_2 \\
\quad | \text{pick-witness } w \text{ for } \exists x . F \Delta \\
\mathcal{D} ::= D | \Delta
\]

where the syntax of inference rule applications is as follows:

\[
\text{RuleApp} ::= \text{claim } F \\
\quad | \text{true-intro} \\
\quad | \text{false-elim} \\
\quad | \text{modus-ponens } F \Rightarrow G, F \\
\quad | \text{modus-tollens } F \Rightarrow G, \neg G \\
\quad | \text{double-negation } \neg \neg F \\
\quad | \text{absurd } F, \neg F \\
\quad | \text{left-and } F \land G
\]
which are read as:

\((\text{Arkoudas 2001a}). \) We illustrate here with the axiom for left-and

\[
\text{left-and } F \land G
\]

both \(F, G\)

left-either \(F, G\)

right-either \(F, G\)

cases \(F_1 \lor F_2, F_1 \Rightarrow G, F_2 \Rightarrow G\)

left-iff \(F \Leftrightarrow G\)

right-iff \(F \Leftrightarrow G\)

equiv \(F \Rightarrow G, G \Rightarrow F\)

The composition operator “;” associates to the right by default, so \(D_1; D_2; D_3\) stands for

\(D_1; (D_2; D_3)\)

rather than \((D_1; D_2); D_3\). Parentheses or begin-end pairs can be used to change the default grouping.

We define \(D[t/x]\) as the deduction obtained from \(D\) by replacing every free occurrence of the variable \(x\) by the term \(t\), taking care to perform \(\alpha\)-conversion as necessary to avoid variable capture. The definition is given by structural recursion:

\[(D_1;D_2)[t/x] = D_1[t/x]; D_2[t/x]\]

\[(\sigma; \rho) \text{ by thinning with } F_1, \ldots, F_n)[t/x] = (\sigma; \rho) \text{ by thinning with } F_1[t/x], \ldots, F_n[t/x]\]

\[
\begin{align*}
(\sigma_1; \rho_1) & \to \Delta_1 | \cdots | (\sigma_n; \rho_n) \to \Delta_n)[t/x] \\
(\text{cases by } F_1, \ldots, F_n; F_1 \to \Delta_1 | F_2 \to \Delta_2)[t/x] & = (\sigma_1; \rho_1) \to \Delta_1[t/x] | \cdots | (\sigma_n; \rho_n) \to \Delta_n[t/x]
\end{align*}
\]

\[
\begin{align*}
(\text{cases } F_1 \lor F_2; F_1 \to \Delta_1 | F_2 \to \Delta_2)[t/x] & = F_1[t/x] \lor F_2[t/x]; \\
\text{pick-witness } x \text{ for } \exists y. F \Delta[t/x] & = \text{pick-witness } x \text{ for } (\exists y. F)[t/x] \Delta \\
\text{pick-witness } w \text{ for } \exists y. F \Delta[t/x] & = \text{pick-witness } w \text{ for } (\exists y. F)[t/x] \Delta[t/x]
\end{align*}
\]

We omit the defining equations for the sentential pick-witness, which is handled like the diagrammatic pick-witness; and for the remaining cases by, which is treated like the one above. The definition for the other forms is straightforward and can be found elsewhere (Arkoudas 2000). In all cases we assume that the deduction has been \(\alpha\)-renamed away from the given term \(t\).

### 1.6.2 Evaluation semantics

Our formal semantics is given by axioms and rules that establish judgments of the form

\[\gamma \vdash D \rightsquigarrow F\]

and

\[\gamma \vdash \Delta \rightsquigarrow (\sigma; \rho)\]

which are read as:

“\(\gamma\) in the context, deduction \(D (\Delta)\) derives \(F\) (respectively, \((\sigma; \rho))\)”

The semantics of most sentential deductions are straightforward generalizations of the standard \(\mathcal{NDL}\) semantics (Arkoudas 2001a). We illustrate here with the axiom for left-and and the rule for assume, omitting the rest:

\[
(\beta \cup \{ F \land G \}; (\sigma; \rho)) \vdash \text{left-and } F \land G \rightsquigarrow F
\]

41
\[
(\beta \cup \{F\}; (\sigma; \rho)) \vdash D \rightarrow G
\]
\[
(\beta; (\sigma; \rho)) \vdash \text{assume } F \ D \rightarrow F \Rightarrow G
\]

In addition, for convenience, we introduce the sentential form

\[ F \text{ by absurdity} \quad (1.99) \]

as syntax sugar for the following sentential deduction:

\[\begin{align*}
\text{assume } & \neg F \\
\text{claim false: } & \quad \text{// This gives } \neg F \Rightarrow \text{false, provided that false is in the assumption base} \\
\text{false-elim: } & \quad \text{// This gives } \neg \text{false} \\
\text{modus-tollens } & \neg F \Rightarrow \text{false, } \neg \text{false; } \quad \text{// Finally, this produces } F
\end{align*}\]

Accordingly, this desugaring ensures that the semantics of (1.99) are as follows:

\[
(\beta \cup \{\text{false}\}; (\sigma; \rho)) \vdash F \text{ by absurdity} \quad (\beta; (\sigma; \rho)) \vdash \text{false} \Rightarrow F
\]

The only new sentential forms (i.e., not present in NDL) are observe, cases by, and \(\Delta; D\). We will discuss the last two later; the semantics of observe are as follows:

\[
(\beta; (\sigma; \rho)) \vdash \text{observe } F \Rightarrow F
\]

provided that \(I_{(\sigma; \rho)/\chi}(F) = \text{true}\) for all \(\chi\)

This rule is used to extract sentential information from diagrams. The side condition is computable because of Lemma 9 and because, by Lemma 7, we only need to be concerned with the free variables of \(F\). In fact usually \(F\) is a sentence (it has no free variables), and hence we only need to consider one—arbitrary—variable assignment.

We now turn to the semantics of the various Vivid constructs for case analysis. There are four types of case reasoning in Vivid:

**Sentential-to-sentential:** In this type of reasoning we note that a disjunction \(F_1 \lor F_2\) holds and that a formula \(G\) is entailed in either case. That entitles us to conclude \(G\). This is captured syntactically as a rule application:

\[
\text{cases } F_1 \lor F_2, F_1 \Rightarrow G, F_2 \Rightarrow G.
\]

The semantics of such rule applications carry over from NDL unchanged, since there is no diagram manipulation involved:

\[
(\beta \cup \{F_1 \lor F_2, F_1 \Rightarrow G, F_2 \Rightarrow G\}; (\sigma; \rho)) \vdash \text{cases } F_1 \lor F_2, F_1 \Rightarrow G, F_2 \Rightarrow G \Rightarrow \text{false} \Rightarrow G
\]

**Sentential-to-diagrammatic:** Here we note that a disjunction \(F_1 \lor F_2\) holds and proceed to show that a certain diagram \((\sigma; \rho)\) follows in either case. This is captured by the syntax form

\[
\text{cases by } F_1 \lor F_2: \quad F_1 \rightarrow \Delta_1 \mid F_2 \rightarrow \Delta_2,
\]

which is classified as a diagrammatic deduction ("a \(\Delta\") since the end result is a diagram. The semantics of this form are given by rule \([C_2]\), shown in Figure 1.11.

**Diagrammatic-to-sentential:** We note that on the basis of the present diagram and some formulas \(F_1, \ldots, F_k\) in the assumption base, \(k \geq 0\), one of \(n > 0\) other system states \((\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)\) must obtain, and proceed to show that a formula \(F\) can be derived in every one of these \(n\) cases. This entitles us to infer \(F\), provided of course that the \(n\) diagrammatic cases are indeed exhaustive. This form of reasoning is captured by the form

\[
\text{cases by } F_1, \ldots, F_k: \quad (\sigma_1; \rho_1) \rightarrow D_1 \mid \cdots \mid (\sigma_n; \rho_n) \rightarrow D_n.
\]
Figure 1.11: Formal semantics of diagrammatic deductions
Likewise, there are four types of deduction sequencing:

1. \(D_1; D_2\), where a sentential deduction \(D_1\) is composed with another sentential deduction \(D_2\). This form is classified as a sentential deduction, since the end result is a formula (the conclusion of \(D_2\)). Its semantics are given by rule \([D; D]\) of Figure [1.1]. They are isomorphic to the regular composition semantics of \(N'D'L\), since there is no diagram manipulation involved.

2. \(D; \Delta\), where a sentential deduction \(D\) is composed with a diagrammatic deduction. This form is classified as a diagrammatic deduction since the end result is a diagram—the conclusion of \(\Delta\). Its semantics are prescribed by rule \([D; \Delta]\). Observe that the conclusion of \(D\) becomes available to \(\Delta\) (e.g., the conclusion of \(D\) could be a disjunction and \(\Delta\) might be a diagrammatic case analysis of that disjunction).

3. \(\Delta; D\), where a diagrammatic deduction \(\Delta\) is composed with a sentential deduction. This form is classified as a sentential deduction since the end result is a formula (the conclusion of \(D\)). Its semantics are given by rule \([\Delta; D]\). Conclusion threading here is also intuitive: \(D\) will be evaluated in the system state resulting from the evaluation of \(\Delta\). E.g., \(D\) might be an observe deduction that points out something that can be seen in the diagram derived by \(\Delta\).

4. \(\Delta_1; \Delta_2\), where a diagrammatic deduction \(\Delta_1\) is composed with another diagrammatic deduction \(\Delta_2\). This form is classified as a diagrammatic deduction, since the end result is a diagram (the conclusion of \(\Delta_2\)). Its semantics are given by rule \([\Delta; \Delta]\). The same principle of conclusion threading applies here: \(\Delta_2\) is evaluated in the system state resulting from the evaluation of \(\Delta_1\); the assumption base is threaded through unchanged.

\[
\begin{align*}
(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_1; \rho_1)) & \vdash D_1 \sim F \\
& \vdots \\
(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_n; \rho_n)) & \vdash D_n \sim F \\
(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) & \vdash \text{cases by } F_1, \ldots, F_k: (\sigma_1; \rho_1) \rightarrow D_1 | \ldots | (\sigma_n; \rho_n) \rightarrow D_n
\end{align*}
\]

Figure 1.12: Semantics of diagrammatic-to-sentential case reasoning.

This is classified as a sentential deduction, since the end result is a formula \(F\). Its semantics are shown in Figure [1.12]. The caveat that the diagrams \((\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)\) form an exhaustive set of possibilities on the basis of \(F_1, \ldots, F_k\) and the current diagram is formally captured by the proviso

\[
(\sigma; \rho) \vdash \{F_1, \ldots, F_k\}\{((\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n))\}.
\]

When \(k = 0\), the by keyword is omitted, and we simply write

\[
\text{cases: } (\sigma_1; \rho_1) \rightarrow D_1 | \cdots | (\sigma_n; \rho_n) \rightarrow D_n.
\]

**Diagrammatic-to-diagrammatic:** This is similar to the above mode of reasoning, with the exception that instead of deriving a formula \(F\) in each of the \(n\) cases, we derive a diagram. Therefore, syntactically, following each of the \(n\) cases we have diagrammatic deductions \(\Delta_1, \ldots, \Delta_n\) (rather than sentential deductions \(D_1, \ldots, D_n\) as we did above), and the entire form is classified as a diagrammatic deduction, since the final conclusion is a diagram. The following syntax form is used for such deductions:

\[
\text{cases by } F_1, \ldots, F_k: (\sigma_1; \rho_1) \rightarrow \Delta_1 | \cdots | (\sigma_n; \rho_n) \rightarrow \Delta_n.
\]

The corresponding semantics are given by rule \([C_1]\), shown in Figure [1.1]. Again, we might have \(k = 0\), and in that case the by keyword is omitted.

Likewise, there are four types of deduction sequencing:

1. \(D_1; D_2\), where a sentential deduction \(D_1\) is composed with another sentential deduction \(D_2\). This form is classified as a sentential deduction, since the end result is a formula (the conclusion of \(D_2\)). Its semantics are given by rule \([D; D]\) of Figure [1.1]. They are isomorphic to the regular composition semantics of \(N'D'L\), since there is no diagram manipulation involved.

2. \(D; \Delta\), where a sentential deduction \(D\) is composed with a diagrammatic deduction. This form is classified as a diagrammatic deduction since the end result is a diagram—the conclusion of \(\Delta\). Its semantics are prescribed by rule \([D; \Delta]\). Observe that the conclusion of \(D\) becomes available to \(\Delta\) (e.g., the conclusion of \(D\) could be a disjunction and \(\Delta\) might be a diagrammatic case analysis of that disjunction).

3. \(\Delta; D\), where a diagrammatic deduction \(\Delta\) is composed with a sentential deduction. This form is classified as a sentential deduction since the end result is a formula (the conclusion of \(D\)). Its semantics are given by rule \([\Delta; D]\). Conclusion threading here is also intuitive: \(D\) will be evaluated in the system state resulting from the evaluation of \(\Delta\). E.g., \(D\) might be an observe deduction that points out something that can be seen in the diagram derived by \(\Delta\).

4. \(\Delta_1; \Delta_2\), where a diagrammatic deduction \(\Delta_1\) is composed with another diagrammatic deduction \(\Delta_2\). This form is classified as a diagrammatic deduction, since the end result is a diagram (the conclusion of \(\Delta_2\)). Its semantics are given by rule \([\Delta; \Delta]\). The same principle of conclusion threading applies here: \(\Delta_2\) is evaluated in the system state resulting from the evaluation of \(\Delta_1\); the assumption base is threaded through unchanged.
We briefly discuss the remaining rules of Figure 1.11. [Thinning] is probably the most frequently used rule for heterogeneous inference in Vivid. It allows us to refine the current state by ruling out worlds that are inconsistent with the cited formulas. [Widening] can be seen as the inverse of thinning, entitling us to “lose information” by increasing rather than decreasing the number of possible worlds that satisfy the current state. This can be useful in getting the diagrammatic branches of a case analysis to be identical. [Absurdity] entitles us to infer any diagram whatsoever from a contradiction. [Diagram-Reiteration] allows us to retrieve the current diagram. [El/Δ] is a diagrammatic version of existential instantiation, whereby we unpack an existential quantification by choosing a witness and then proceed with a diagrammatic deduction.

**Theorem 28 (Soundness):** If \( \gamma \vdash D \leadsto F \) then \( \gamma \models F \); and if \( \gamma \vdash \Delta \leadsto (\sigma; \rho) \) then \( \gamma \models (\sigma; \rho) \).

**Proof:** We proceed by induction on derivation length.\(^{28}\) We will omit most sentential forms, as those have been proved sound elsewhere (Arkoudas 2000).

The basis cases correspond to the axioms of our semantics. Here we treat the diagrammatic axioms [Observe], [Absurdity], [Diagram-Reiteration], [Widening], and [Thinning].

- **[Observe]:** In this case \( D \) is of the form \( \text{observe } F \) and we need to show that
  \[
  (\beta; (\sigma; \rho)) \models F
  \]
  whenever \((\beta; (\sigma; \rho)) \vdash D \leadsto F\). To that end, consider an arbitrary world \((w; \rho)\) and variable assignment \(\chi\) and suppose that \((w; \rho) \models \chi (\beta; (\sigma; \rho))\), so that
  \[
  (w; \rho) \subseteq (\sigma; \rho). \tag{1.100}
  \]
  By the side condition of [Observe], it must be that
  \[
  I_{(\sigma; \rho)/\chi}(F) = \text{true},
  \]
  and hence, from (1.100) and Lemma 8
  \[
  \forall_{(w; \rho)/\chi} F = \text{true},
  \]
  which is to say \((w; \rho) \models \chi F\). We have thus shown that \((w; \rho) \models \chi (\beta; (\sigma; \rho))\) implies \((w; \rho) \models \chi F\) for any \((w; \rho)\) and \(\chi\), which establishes \((\beta; (\sigma; \rho)) \models F\).

- **[Thinning]:** Here \( \Delta \) is of the form \( (\sigma'; \rho') \) by thinning with \( F_1, \ldots, F_n \), and we need to show that if
  \[
  (\beta \cup \{F_1, \ldots, F_n\}; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho') \tag{1.101}
  \]
  then
  \[
  (\beta \cup \{F_1, \ldots, F_n\}; (\sigma; \rho)) \models (\sigma'; \rho'). \tag{1.102}
  \]
  From (1.101) and the side condition of [Thinning] we obtain
  \[
  (\sigma; \rho) \models (\sigma'; \rho'),
  \]
  and hence, by Corollary 26 \( (\{F_1, \ldots, F_n\}; (\sigma; \rho)) \models (\sigma'; \rho') \). Now (1.102) follows from weakening (Lemma 17).

- **[Widening]:** Here \( \Delta \) is of the form \( (\sigma'; \rho') \) by widening and we must show that \((\beta; (\sigma; \rho)) \models (\sigma'; \rho')\) whenever \((\beta; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho')\). From the side condition of [Widening] we infer \((\sigma; \rho) \subseteq (\sigma'; \rho')\), and now the desired \((\beta; (\sigma; \rho)) \models (\sigma'; \rho')\) follows from Corollary 22.

- **[Diagram-Reiteration]:** Here the result follows directly from Lemma 19.

\(^{28}\)To be perfectly precise, we are proving the statement: “For all positive integers \( n \) and for all \( \gamma, D, \Delta, F, \) and \( (\sigma; \rho) \), if there exists a derivation of length \( n \) of the judgment \( \gamma \vdash D \leadsto F \) then \( \gamma \models F \); and if there exists a derivation of length \( n \) of \( \gamma \vdash \Delta \leadsto (\sigma; \rho) \) then \( \gamma \models (\sigma; \rho) \).” It is easy to see that this statement implies Theorem 28.
• [Absurdity]: Here $\Delta$ is of the form

$(\sigma'; \rho')$ by absurdity

and we need to show

$$(\beta \cup \{\text{false}\}; (\sigma; \rho)) \vdash (\sigma'; \rho')$$

whenever $(\beta \cup \{\text{false}\}; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho')$. This follows from Lemma 20.

The inductive steps correspond to the proper evaluation rules of our semantics. In what follows we consider every possible case.

• [C1]: Here $\Delta$ is of the form

**cases** by $F_1, \ldots, F_k$: $(\sigma_1; \rho_1) \rightarrow \Delta_1 \mid \cdots \mid (\sigma_n; \rho_n) \rightarrow \Delta_n$.

Consider any assumption base $\beta$ and named states $(\sigma; \rho), (\sigma'; \rho')$, and assume that

$$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho').$$

We need to show

$$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \vdash (\sigma'; \rho').$$

From (1.103) and [C1] we infer

$$\forall i \in \{1, \ldots, n\}. (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_i; \rho_i)) \vdash \Delta_i \leadsto (\sigma_i'; \rho_i')$$

and

$$(\sigma; \rho) \vDash \{F_1, \ldots, F_k\} \{(\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)\}.$$  

(1.106)

Pick any world $(w; \bar{\rho})$ and variable assignment $\chi$ and suppose that

$$(w; \bar{\rho}) \vDash \chi (\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)),$$

so that

$$(w; \bar{\rho}) \vDash \chi (\{F_1, \ldots, F_k\}; (\sigma; \rho)).$$

(1.108)

From (1.106), Lemma 25, and (1.108), we conclude that $(w; \bar{\rho}) \vDash (\sigma_j; \rho_j)$ for some $j \in \{1, \ldots, n\}$. By the inductive hypothesis, (1.105) yields

$$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_j; \rho_j)) \vdash (\sigma'; \rho'),$$

(1.109)

and since

$$(w; \bar{\rho}) \vDash \chi (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_j; \rho_j)),$$

it follows from (1.109) that $(w; \bar{\rho}) \vDash (\sigma'; \rho')$.

• [C2]: Here $\Delta$ is of the form

**cases** $F_1 \lor F_2$: $F_1 \rightarrow \Delta_1 \mid F_2 \rightarrow \Delta_2$

and, assuming

$$(\beta \cup \{F_1 \lor F_2\}; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho'),$$

we need to show

$$(\beta \cup \{F_1 \lor F_2\}; (\sigma; \rho)) \vdash (\sigma'; \rho').$$

(1.111)

To that end, consider an arbitrary world $(w; \bar{\rho})$ and variable assignment $\chi$ such that

$$(w; \bar{\rho}) \vDash \chi (\beta \cup \{F_1 \lor F_2\}; (\sigma; \rho))$$

(1.112)

so that

$$V^I_{(w; \bar{\rho})/\chi}[F_1] = \text{true}$$

(1.113)
or

$$\forall^I_{(w;\tilde{\rho})/\chi}[F_2] = \text{true.}$$

(1.114)

Now from (1.110) and [C2] we get

$$(\beta \cup \{F_1 \lor F_2, F_1\}; (\sigma; \rho)) \vdash \Delta_1 \backsim (\sigma'; \rho')$$

(1.115)

and

$$(\beta \cup \{F_1 \lor F_2, F_2\}; (\sigma; \rho)) \vdash \Delta_2 \backsim (\sigma'; \rho').$$

(1.116)

Inductively, (1.115) and (1.116) respectively yield

$$(\beta \cup \{F_1 \lor F_2, F_1\}; (\sigma; \rho)) \models (\sigma'; \rho')$$

(1.117)

and

$$(\beta \cup \{F_1 \lor F_2, F_2\}; (\sigma; \rho)) \models (\sigma'; \rho').$$

(1.118)

Now if (1.113) holds then, from (1.112), we have

$$(w; \tilde{\rho}) \models_\chi (\beta \cup \{F_1 \lor F_2, F_1\}; (\sigma; \rho)),$$

and hence $$(w; \tilde{\rho}) \models (\sigma'; \rho')$$ follows from (1.117); while if (1.114) holds then

$$(w; \tilde{\rho}) \models_\chi (\beta \cup \{F_1 \lor F_2, F_2\}; (\sigma; \rho)),$$

and hence $$(w; \tilde{\rho}) \models (\sigma'; \rho')$$ follows from (1.118). Therefore, $$(w; \tilde{\rho}) \models (\sigma'; \rho')$$ holds in either case.

• [C3]: Here $\Delta$ is of the form

cases by $F_1, \ldots, F_k$: $(\sigma_1; \rho_1) \rightarrow D_1 \mid \cdots \mid (\sigma_n; \rho_n) \rightarrow D_n$.

Pick any $\beta, F$, and $(\sigma; \rho)$, and suppose that

$$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \vdash \Delta \backsim F;$$

(1.119)

so that

$$\forall i \in \{1, \ldots, n\}. (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_i; \rho_i)) \vdash D_i \backsim F$$

(1.120)

and

$$(\sigma; \rho) \mid\mid \{F_1, \ldots, F_k\} \{((\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n))\}.$$ (1.121)

We need to show $$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \models F.$$ To that end, pick any $$(w; \tilde{\rho})$$ and $\chi$$ and assume that

$$(w; \tilde{\rho}) \models_\chi (\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)).$$

(1.122)

It follows that

$$(w; \tilde{\rho}) \models_\chi (\{F_1, \ldots, F_k\}; (\sigma; \rho)),$$

and hence by Lemma [25] and (1.121) we conclude that $$(w; \tilde{\rho}) \subseteq (\sigma_j; \rho_j)$$ for some $j \in \{1, \ldots, n\}$. Inductively, from (1.120), we infer

$$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_j; \rho_j)) \models F.$$ (1.123)

But from $$(w; \tilde{\rho}) \subseteq (\sigma_j; \rho_j)$$ and (1.122) we get

$$(w; \tilde{\rho}) \models_\chi (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_j; \rho_j)),$$

and therefore (1.123) yields $$(w; \tilde{\rho}) \models_\chi F.$$
\[ [El/\Delta]: \text{In that case the deduction is of the form} \]

\[ \text{pick-witness } w \text{ for } \exists x . F \quad \Delta \]

and assuming that

\[ (\beta \cup \{ \exists x . F \}; (\sigma; \rho)) \vdash \text{pick-witness } w \text{ for } \exists x . F \vdash (\sigma'; \rho'), \quad (1.124) \]

we need to show

\[ (\beta \cup \{ \exists x . F \}; (\sigma; \rho)) \models (\sigma'; \rho'). \quad (1.125) \]

To that end, consider any \((w; \widehat{\rho})\) and \(\chi\) such that

\[ (w; \widehat{\rho}) \models \chi (\beta \cup \{ \exists x . F \}; (\sigma; \rho)). \quad (1.126) \]

From \((1.124)\) and the \([El/\Delta]\) rule we infer that, for some fresh variable \(z\),

\[ (\beta \cup \{ \exists x . F, F[z/x] \}; (\sigma; \rho)) \vdash \Delta [z/w] \leadsto (\sigma'; \rho'). \quad (1.127) \]

From \((1.127)\) and the inductive hypothesis we obtain

\[ (\beta \cup \{ \exists x . F \}; (\sigma; \rho)) \models (\sigma'; \rho'). \quad (1.128) \]

From \((1.126)\) and \((1.35)\) we conclude that there is some system object \(s\) such that

\[ \mathcal{V}^I_{(w; \widehat{\rho})/\chi [x \mapsto s]}[F] = \text{true}. \]

Therefore, from Lemma \(27\)

\[ \mathcal{V}^I_{(w; \widehat{\rho})/\chi [z \mapsto s]}[F[z/x]] = \text{true}, \]

and since \(z\) does not occur in \(\beta \cup \{ \exists x . F \}\), we also have (by \((1.126)\) and Lemma \(7\)):

\[ \forall G \in \beta \cup \{ \exists x . F \}. \mathcal{V}^I_{(w; \widehat{\rho})/\chi [z \mapsto s]}[G] = \text{true}. \]

Hence,

\[ (w; \widehat{\rho}) \models \chi [z \mapsto s] (\beta \cup \{ \exists x . F, F[z/x] \}; (\sigma; \rho)). \quad (1.129) \]

and since \((w; \widehat{\rho}) \subseteq (\sigma; \rho)\) (from \((1.126)\)), we conclude that

\[ (w; \widehat{\rho}) \models \chi [z \mapsto s] (\beta \cup \{ \exists x . F \}; (\sigma; \rho)). \quad (1.130) \]

Finally, from \((1.130)\) and \((1.128)\) we obtain \((w; \widehat{\rho}) \models (\sigma'; \rho'). \]

\[ [D; \Delta]: \text{Here the deduction is of the form } D; \Delta, \text{ and assuming that} \]

\[ (\beta; (\sigma; \rho)) \vdash D; \Delta \vdash (\sigma'; \rho'), \quad (1.131) \]

we need to show

\[ (\beta; (\sigma; \rho)) \models (\sigma'; \rho'). \quad (1.132) \]

Pick any \((w; \widehat{\rho})\) and \(\chi\) and suppose that

\[ (w; \widehat{\rho}) \models \chi (\beta; (\sigma; \rho)). \quad (1.133) \]

From \((1.131)\) and the \([D; \Delta]\) rule we infer that, for some \(F\),

\[ (\beta; (\sigma; \rho)) \vdash D \leadsto F; \quad (1.134) \]

and since \((\beta \cup \{ F \}; (\sigma; \rho)) \models (\sigma'; \rho'). \)

\[ (\beta \cup \{ F \}; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho'). \quad (1.135) \]

From \((1.134)\) and the inductive hypothesis we obtain \((\beta; (\sigma; \rho)) \models F\), which, in tandem with \((1.133)\), yields

\[ (w; \widehat{\rho}) \models \chi F. \quad (1.136) \]

Therefore,

\[ (w; \widehat{\rho}) \models \chi (\beta \cup \{ F \}; (\sigma; \rho)). \quad (1.137) \]

Now \((1.135)\) and the inductive hypothesis give

\[ (\beta \cup \{ F \}; (\sigma; \rho)) \models (\sigma'; \rho'), \quad (1.137) \]

and finally \((1.136)\) and \((1.137)\) produce the desired \((w; \widehat{\rho}) \models (\sigma'; \rho'). \)
• \([\Delta; D]\): Here the proof is of the form \(\Delta; D\) and assuming that
\[
(\beta; (\sigma; \rho)) \vdash \Delta; D \leadsto F,
\]
we need to show \((\beta; (\sigma; \rho)) \models F\). Accordingly, consider any world \((w; \hat{\rho})\) and variable assignment \(\chi\) such that
\[
(w; \hat{\rho}) \models \chi (\beta; (\sigma; \rho)).
\]
From (1.138) and the \([\Delta; D]\) rule we conclude that, for some \((\sigma'; \rho')\),
\[
(\beta; (\sigma; \rho)) \vdash \Delta \leadsto (\sigma'; \rho')
\]
and
\[
(\beta; (\sigma'; \rho')) \vdash D \leadsto F.
\]
From (1.140) and the inductive hypothesis we get \((\beta; (\sigma; \rho)) \models (\sigma'; \rho')\), so (1.139) yields
\[
(w; \hat{\rho}) \models (\sigma'; \rho')
\]
and hence
\[
(w; \hat{\rho}) \models \chi (\beta; (\sigma'; \rho')).
\]
But from (1.141) and the inductive hypothesis we get \((\beta; (\sigma'; \rho')) \models F\), which, along with (1.142), entails
\[
(w; \hat{\rho}) \models \chi F.
\]

• \([\Delta; \Delta]\): Here the deduction is of the form \(\Delta; \Delta\). Assuming
\[
(\beta; (\sigma; \rho)) \vdash \Delta_1; \Delta_2 \leadsto (\sigma_2; \rho_2),
\]
we must show \((\beta; (\sigma; \rho)) \models (\sigma_2; \rho_2)\). Pick any \((w; \hat{\rho})\) and \(\chi\) such that
\[
(w; \hat{\rho}) \models \chi (\beta; (\sigma; \rho)).
\]
From (1.143) and rule \([\Delta; \Delta]\) we infer that, for some \((\sigma_1; \rho_1)\),
\[
(\beta; (\sigma; \rho)) \vdash \Delta_1 \leadsto (\sigma_1; \rho_1);
\]
\[
(\beta; (\sigma_1; \rho_1)) \vdash \Delta_2 \leadsto (\sigma_2; \rho_2).
\]
From (1.145), (1.146), and the inductive hypotheses we get
\[
(\beta; (\sigma; \rho)) \models (\sigma_1; \rho_1);
\]
\[
(\beta; (\sigma_1; \rho_1)) \models (\sigma_2; \rho_2).
\]
From (1.144) and (1.147) we infer \((w; \hat{\rho}) \models (\sigma_1; \rho_1)\), so that
\[
(w; \hat{\rho}) \models \chi (\beta; (\sigma_1; \rho_1)),
\]
which in tandem with (1.148) yields the desired \((w; \hat{\rho}) \models (\sigma_2; \rho_2)\).

This completes the case analysis and the induction.

Example 13: Consider the Vivid language obtained by fixing the clock signature, attribute structure and interpretation of Example 11. Now consider a system of two clocks \(s_1\) and \(s_2\), to which we will give the names \(c_1\) and \(c_2\) (recall that \(c_1\) and \(c_2\) are constant symbols of the signature, so this is a constant assignment \(\rho\), which need only be partial). Now let \(\sigma\) be the state depicted by the following picture:

\[
\begin{array}{c}
\{4, 5, 6\}: 28 \\
c_1
\end{array}
\quad
\begin{array}{c}
5: 45 \\
c_2
\end{array}
\]
Intuitively, this state signifies that we know the precise time displayed by \( s_2 \) (5:45 am). We are also sure of the minute value of \( s_1 \) (28), but not of its hour value, which could be either 4, 5, or 6. Now suppose that we are further given the premise \( \text{Ahead}(c_1, c_2) \), indicating that the time displayed by \( s_1 \) is ahead of that displayed by \( s_2 \).

From these two pieces of information, one diagrammatic and the other sentential, we should be able to infer the following diagram, call it \( \sigma' \):

![Diagram](attachment:image.png)

That is, we should be able to conclude the exact time of \( s_1 \), since, given that \( s_1 \) is ahead of \( s_2 \), the hour displayed by it cannot possibly be 4 or 5; it must, therefore, be 6. We can do this in Vivid with the following one-line proof:

\[(\sigma'; \rho) \text{ by thinning with } \text{Ahead}(c_1, c_2).\]

This deduction, when evaluated in the context \( \{\text{Ahead}(c_1, c_2); (\sigma; \rho)\} \), will result in the state (diagram) \( (\sigma'; \rho) \). More formally, we have the following judgment:

\[(\{\text{Ahead}(c_1, c_2); (\sigma; \rho)\} \vdash (\sigma'; \rho) \text{ by thinning with } \text{Ahead}(c_1, c_2) \sim \sigma' \rho)\]

by virtue of

\[(\sigma; \rho) \vdash (\sigma'; \rho).\]  \hspace{1cm} (1.149)

Note that \( \rho \) does not change in the resulting state.

To establish (1.149) rigorously, we must show that for all named states \( (\sigma''; \rho'') \) such that

\[\text{Alt}(\sigma; \rho, (\sigma'; \rho), (\sigma''; \rho''))\]

we have

\[\mathcal{V}(\sigma''; \rho'')/\chi(\text{Ahead}(c_1, c_2)) = \text{false}\]

for all variable assignments \( \chi \), according to Definition 12. Given that the constant assignment does not change, it follows from Definition 11 that we must have \( \rho'' = \rho \) and hence \( \text{Alt}(\sigma, \sigma', \sigma'') \). Now there is only one alternative extension \( \sigma'' \) of \( \sigma \) w.r.t. \( \sigma' \), obtained from \( \sigma \) by complementing the hours value of \( s_1 \) in \( \sigma' \) with respect to the corresponding value in \( \sigma \):

\[
\begin{align*}
\sigma''(\text{hours, } s_1) &= \{4, 5\}; \\
\sigma''(\text{minutes, } s_1) &= \{28\}; \\
\sigma''(\text{hours, } s_2) &= \{5\}; \\
\sigma''(\text{minutes, } s_2) &= \{45\}.
\end{align*}
\]

It is straightforward to verify that

\[\mathcal{V}(\sigma''; \rho)/\chi(\text{Ahead}(c_1, c_2)) = \text{false}\]

for all \( \chi \).

\[\blacksquare\]

### 1.7 Representing arbitrary graphs

Graphs (including trees, lists, etc.) are very widely used as diagrammatic depictions of structured data. In this section we present a way of modeling arbitrary graphs in our framework as system states. These ideas will be put to use in the example of Section 1.8.

Consider an arbitrary finite graph \( G = (N; E) \), where \( N \) is a set of nodes and \( E \subseteq N \times N \) a set of directed edges. Typically we wish to attach a value to each node \( n \in N \), so we assume we have a function \( \text{data} : N \to V \) that maps each node to some element of a set of values \( V \). For the purposes of drawing the graph, we also assume that the children of every node are ordered from left to right, i.e., we assume there is a function \( \text{children} : N \to N^* \) (arbitrary lists can be chosen if the ordering is immaterial for displaying the graph). Consider, for instance, the graph:
Figure 1.13: The call graph resulting from the application of Mergesort to the list [5 8 3 2].

Here \( N = \{n_1, n_2, n_3\} \) and \( E = \{(n_1, n_2), (n_1, n_3)\} \). The values attached to the nodes are natural numbers. So we can represent the graph by the functions \( \text{data} \) and \( \text{children} \) as mentioned above, where

\[
\text{data}(n_1) = 5, \text{data}(n_2) = 3, \text{data}(n_3) = 3
\]

and

\[
\text{children}(n_1) = [n_2, n_3], \text{children}(n_2) = [], \text{children}(n_3) = [].
\]

This is similar to the adjacency-list representation of graphs (Cormen, Leiserson and Rivest 1990).

Any graph \( G = (N; E) \) where the nodes take values from a set \( V \) gives rise to systems of the form \( S_N = (N; A_N) \), where \( A_N \) is an automorphic attribute structure of the form

\[
A_N = (id : N, \text{children} : N^*, \text{data} : V; R).
\]

Here the attributes \( \text{children} \) and \( \text{data} \) are as discussed above, \( id \) is the identity function on \( N \), and \( D(R) \subseteq \{N, N^*, V\} \) for each relation \( R \in R \) (the precise contents of \( R \) will vary). The graph \( G \) itself can be represented as a world of the system \( S_N \). “Incomplete” graphs where the values and/or children of some nodes are not precisely known can be represented by partial states of such systems.

1.8 Another example: the Mergesort puzzle

In this section we present a more involved Vivid language by way of a puzzle. In its general form, the puzzle can be described as follows. The output of an algorithm is displayed at the bottom of a diagram depicting a call graph for a particular run of the algorithm. Some sentential information might also be given in addition to the diagram. The objective is to infer what input(s) could possibly have resulted in the given call graph, or, more precisely, what inputs are consistent with the given information (the call graph and the sentences). Inference is mostly performed diagrammatically, by deriving a sequence of successive call graphs, by performing case analyses involving such graphs, etc. It will be seen that such graphical proofs are considerably more compact and intuitive than sentential analogues. In the next section we illustrate the puzzle informally with Mergesort, while in Section 1.8.2 we formalize it rigorously as a Vivid system.

1.8.1 Guessing the input of Mergesort

Mergesort is a popular \( O(n \log n) \) sorting algorithm. The algorithm works according to the divide-and-conquer paradigm (Cormen et al. 1990): it successively halves the given list until the original input has been broken into one-element pieces,
which are trivially sorted; this is the dividing phase. The small lists are then repeatedly combined into larger and larger sorted lists, until we finally obtain the correct sorted permutation of the original input. This is the conquering phase, which turns on the fact that once we have two sorted lists, say [2 8] and [1 3 5], we can efficiently merge them to get another sorted list, in this case [1 2 3 5 8].

For example, Figure 1.13 depicts the call graph obtained by applying Mergesort to the input list [5 8 3 2]. Note that the graph is a DAG (directed acyclic graph). Diverging edges on the top half represent recursive applications of Mergesort to the left and right halves of the input (dividing phase); while converging edges on the lower half represent calls to the merging procedure (conquering phase). We make the convention that when the input list is of an odd length \(2n+1\), we take the first \(n\) elements as the left half and the remaining \(n+1\) elements as the right half.

The call graph for an application of Mergesort is completely and unambiguously determined once the input list is given. However, things are more interesting in the reverse direction. Clearly, there is no way of retrieving the input list from the output alone, since the inverse of a sorting function is a relation, not a function—any one of \(n!\) initial permutations could result in the same sorted \(n\)-element list. But if, in addition to specifying the output, we also constrain the call graph of the algorithm by sprinkling some tidbits of information on it or by specifying some sentential information along with it, then we may be able to infer the original input, or at least narrow it down to relatively few possibilities.

As a simple example, suppose you are told that the output of Mergesort is [1 2 5 8]. At this point there is not much of interest you can conclude—there are \(4! = 24\) possible inputs that could produce this output. But suppose you are further told that the corresponding call graph is as shown in Figure 1.14, where we have attached labels \(N_i\) to each node of the graph for easy reference. We write \(N_i = ?\) to indicate that we do not know anything about the list that should appear at node \(N_i\); we write \(N_i \supseteq \{x_1, \ldots, x_k\}\) to indicate that the numbers \(x_1, \ldots, x_k\) occur in the said list (though in unknown order, and possibly in tandem with other numbers); and \(N_i = [x_1 \cdots x_k]\) to indicate that we know the exact value of the list in question to be \([x_1 \cdots x_k]\). From the diagram of Figure 1.14 along with what we know about the way Mergesort operates, we can conclude that the original input was either [2 5 8 1] or [5 2 8 1].

The proof consists of two parts. First we derive a sequence of six increasingly detailed diagrams from the initial diagram of Figure 1.14, each extending the previous one, culminating with a diagram in which we know the exact values of all the lists except those for \(N_1, N_2, N_4\) and \(N_5\); this part of the proof appears in Figure 1.15. We then perform an exhaustive case analysis by observing that there are only two possibilities at this point: the lists of \(N_4\) and \(N_5\) are (a) [2] and [5], respectively; or else they are (b) [5] and [2], respectively. In the first case we can deduce that the input list was [2 5 8 1], while in the second case we can deduce that it was [5 2 8 1]. Therefore, we can infer that the input list was either [2 5 8 1] or [5 2 8 1].

Let us analyze the proof in more detail, beginning with the first part shown in Figure 1.15. That part consists of six steps, labeled (1) through (6). The new information extracted by each step is underlined for enhanced clarity. We discuss each step below:

- Step (1) infers that \(N_6\) must contain the number 8. This follows because we know that 8 occurs in \(N_9\) but not in

\[N_9 = ? \quad N_6 = ? \quad N_7 = [1] \quad N_8 = ? \quad N_9 \supseteq \{8\}\]

\[N_{10} = [1 2 5 8] \]

Figure 1.14: A partially unknown Mergesort call graph resulting in the output [1 2 5 8].
Figure 1.15: First part of a graphical proof solving an instance of the Mergesort puzzle.
\(N_7\); and that, since \(N_6\) and \(N_7\) converge in \(N_9\), a number can occur in \(N_9\) iff it occurs either in \(N_6\) or in \(N_7\) (this holds because converging edges indicate list merging).

- Step (2) infers that the list appearing in node \(N_6\) must be precisely \([8]\). We already know from the previous step that 8 occurs in the said list. Now if the list had any additional elements, its length would be greater than one, and hence it would be longer than the \(N_7\) list, which we know to have only one element. But this cannot be the case because \(N_6\) and \(N_7\) are the left and right halves of the \(N_3\) list, and every time a list \(L\) is split into two halves, the left half is always either of the same length as the right half (if \(L\) has even length) or else it is shorter by one (if \(L\) has odd length); it cannot possibly be longer. Hence, the \(N_6\) list must be the one-element list \([8]\).

- Step (3) infers that the \(N_9\) list must be \([1\ 8]\). This follows because the \(N_9\) list represents the result of merging \(N_6\) and \(N_7\), whose precise values are both known at this point.

- Step (4) infers that the \(N_3\) list must be \([8\ 1]\). This follows because we already know the left and right halves of \(N_3\) to be \([8]\) and \([1]\), respectively.

- Step (5) infers that the \(N_8\) list must contain 2 and 5. This holds by virtue of the principle mentioned above in connection with step (1): when \(L\) and \(L'\) converge in \(L''\), any number occurs in \(L''\) iff it occurs either in \(L\) or in \(L'\). Therefore, since we know that 2 and 5 occur in \(N_{10}\) but not in \(N_9\), they must occur in \(N_8\).

- Step (6) infers that the \(N_8\) list must be precisely \([2\ 5]\). We already know that it must have at least these two elements. If it had more than two elements, then \(N_{10}\) would have to have at least five elements, given that (a) \(N_{10}\)
is the result of merging $N_8$ and $N_9$, and that (b) $N_9$ has two elements. But $N_{10}$ has four elements, therefore 2 and 5 must be the only two elements of $N_8$, leaving [2 5] and [5 2] as the only two possibilities. But the second possibility cannot hold, since $N_8$ must be sorted (recall that only sorted lists get merged). Hence, the $N_8$ list must be [2 5].

At this point we do not have sufficient information to determine unique values for the $N_4$ and $N_5$ lists. However, we can narrow things down to two possibilities: either $N_4$ and $N_5$ are [2] and [5], respectively; or else they are [5] and [2]. These are the only two alternatives that are consistent with $N_8$ = [2 5], given that $N_8$ represents the result of merging $N_4$ and $N_5$. The reasoning in each case is as follows:

**Case 1**: In that case (Figure 1.16), we proceed to infer that the value of $N_2$ must be [2 5], since $N_4$ and $N_5$ are the left and right halves of $N_2$. And then, since we know both $N_2$ and $N_3$ we can determine the value of the input $N_1$ to be [2 5 8 1].

**Case 2**: In that case (Figure 1.17), we deduce that the value of $N_2$ must be [5 2], for the same reason we cited in the preceding case. Similarly, we can then conclude that the input list must be [5 2 8 1].

We are now entitled to infer that the original input list must be either [2 5 8 1] or [5 2 8 1].

**1.8.2 Formalizing the puzzle as a Vivid system**

There are three steps to obtaining a particular instance of Vivid:
The structure for the Mergesort puzzle is the following:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Arity</th>
<th>Realization</th>
</tr>
</thead>
<tbody>
<tr>
<td>peak</td>
<td>3</td>
<td>$R_1$</td>
</tr>
<tr>
<td>valley</td>
<td>3</td>
<td>$R_2$</td>
</tr>
<tr>
<td>append</td>
<td>3</td>
<td>$R_3$</td>
</tr>
<tr>
<td>halves</td>
<td>2</td>
<td>$R_4$</td>
</tr>
<tr>
<td>sorted</td>
<td>1</td>
<td>$R_5$</td>
</tr>
<tr>
<td>union</td>
<td>3</td>
<td>$R_6$</td>
</tr>
<tr>
<td>sum</td>
<td>3</td>
<td>$R_7$</td>
</tr>
<tr>
<td>$val_L$</td>
<td>1</td>
<td>$R_L$</td>
</tr>
</tbody>
</table>

In the following three sections we carry out these steps in detail for the Mergesort puzzle.

### Specifying the attribute structure

Let $Node$ be the universe of nodes and let $Z^*$ be the set of all finite sequences (lists) of integers. An appropriate attribute structure for the Mergesort puzzle is the following:

$$A_M = \{id : Node, children : Node^*, data : Z^*; \{R_1, R_2, R_3, R_4, R_5, R_6, R_7\} \cup \{R_L \mid L \in Z^*\}\}$$

where the relations $R_1, \ldots, R_7, R_L$ are as follows:

1. $R_1 \subseteq Node^* \times Node \times Node$, with
   $$R_1([n_1 \ldots n_k], n, n') \iff \{n, n'\} \subseteq \{n_1, \ldots, n_k\}.$$

2. $R_2 \subseteq Node \times Node^* \times Node^*$, with
   $$R_2(n, [n_1 \ldots n_k], [n'_1 \ldots n'_m]) \iff n \in \{n_1, \ldots, n_k\} \cap \{n'_1, \ldots, n'_m\}.$$

3. $R_3 \subseteq Z^* \times Z^* \times Z^*$, with
   $$R_3([x_1 \ldots x_k], [y_1 \ldots y_n], [z_1 \ldots z_m]) \iff [x_1 \ldots x_k] = [y_1 \ldots y_n, z_1 \ldots z_m],$$
   i.e., iff $[x_1 \ldots x_k]$ is the concatenation of $[y_1 \ldots y_n]$ and $[z_1 \ldots z_m]$.

4. $R_4 \subseteq Z^* \times Z^*$, with
   $$R_4([x_1 \ldots x_k], [y_1 \ldots y_n]) \iff n \in \{k, k + 1\}.$$

5. $R_5 \subseteq Z^*$, with
   $$R_5([x_1 \ldots x_k]) \iff x_i \leq x_{i+1} \text{ for } i = 1, \ldots, k - 1,$$
   i.e., iff $[x_1 \ldots x_k]$ is sorted.

6. $R_6 \subseteq Z^* \times Z^* \times Z^*$, with
   $$R_6([x_1 \ldots x_k], [y_1 \ldots y_n], [z_1 \ldots z_m]) \iff \{x_1, \ldots, x_k\} = \{y_1, \ldots, y_n\} \cup \{z_1, \ldots, z_m\}.$$

7. $R_7 \subseteq Z^* \times Z^* \times Z^*$, with
   $$R_7([x_1 \ldots x_k], [y_1 \ldots y_n], [z_1 \ldots z_m]) \iff k = n + m.$$
Figure 1.19: SML code implementing the attribute structure of the Mergesort puzzle.

- \( R_L \subseteq \mathbb{Z}^* \), with
  \[
  R_{[x_1 \ldots x_k]}([y_1 \ldots y_n]) \iff [x_1 \ldots x_k] = [y_1 \ldots y_n].
  \]

Note that we have infinitely many unary relations \( R_L \), parameterized by \( L \). Each such relation takes an arbitrary list of integers \( L' \) and tests for the equality \( L' = L \).

To make things concrete, Figure 1.19 presents an implementation of this attribute structure in SML.

**Specifying the vocabulary**

We have seven relations symbols: peak, valley, append, union, and sum are ternary; halves is binary; and sorted is unary. In addition, for each list of integers \( L \) we have a unary relation symbol \( \text{val}_L \). We use \( N_1, N_2, \ldots \) as constant symbols and \( v_1, v_2, \ldots \) as variables.

**Specifying the interpretation**

The interpretation of the relation symbols is shown in Figure 1.18.

More intuitive explanations follow:

- \( \text{peak}(v_1, v_2, v_3) \) holds iff nodes \( v_2 \) and \( v_3 \) are both children of \( v_1 \):

- \( \text{valley}(v_1, v_2, v_3) \) holds iff \( v_2 \) and \( v_3 \) are both parents of \( v_1 \):
append\((v_1, v_2, v_3)\) holds iff the list attached to node \(v_1\) (i.e., the data field of \(v_1\)) is identical to the concatenation of the lists attached to nodes \(v_2\) and \(v_3\), respectively.

halves\((v_1, v_2)\) holds iff the lengths of the lists attached to nodes \(v_1\) and \(v_2\) are approximately equal; more precisely, iff the length of the \(v_2\) list is either equal to or one more than the length of the \(v_1\) list.

sorted\((v_1)\) holds iff the list attached to node \(v_1\) is sorted.

union\((v_1, v_2, v_3)\) holds iff the list attached to node \(v_1\) contains all and only those elements that occur either in \(v_2\) or in \(v_3\) (or in both).

sum\((v_1, v_2, v_3)\) holds iff the length of the \(v_1\) list is equal to the sum of the lengths of the \(v_2\) and \(v_3\) lists.

val\(_L\)(\(v_1\)) holds iff the list attached to node \(v_1\) is identical to \(L\). We write val\((v_1, L)\) as an abbreviation for val\(_L\)(\(v_1\)).

1.8.3 The formal proof

The following Horn clauses are all the axioms we need for solving Mergesort puzzles. Their meaning should be clear in light of the foregoing interpretation.

\[
\begin{align*}
\forall v_1, v_2, v_3. \text{valley}(v_1, v_2, v_3) & \lor \text{peak}(v_1, v_2, v_3) \Rightarrow \text{halves}(v_2, v_3) & \text{halves-axiom} \\
\forall v_1, v_2, v_3. \text{valley}(v_1, v_2, v_3) & \Rightarrow \text{sorted}(v_1) \land \text{sorted}(v_2) \land \text{sorted}(v_3) & \text{sorted-axiom} \\
\forall v_1, v_2, v_3. \text{valley}(v_1, v_2, v_3) & \lor \text{peak}(v_1, v_2, v_3) \Rightarrow \text{union}(v_1, v_2, v_3) & \text{union-axiom} \\
\forall v_1, v_2, v_3. \text{peak}(v_1, v_2, v_3) & \Rightarrow \text{append}(v_1, v_2, v_3) & \text{append-axiom} \\
\forall v_1, v_2, v_3. \text{valley}(v_1, v_2, v_3) & \Rightarrow \text{sum}(v_1, v_2, v_3) & \text{sum-axiom}
\end{align*}
\]

Now let node\(_1\), …, node\(_{10}\) be ten nodes from the universe of all nodes, Node. In combination with the attribute structure \(A_M\), these ten nodes constitute a system. The diagrams shown in Figure 1.15, Figure 1.16, and Figure 1.17 depict specific named states of this system. Consider, for instance, the starting diagram, at the upper left corner of Figure 1.15. This represents a named state \((\sigma; \rho)\), where the partial constant assignment \(\rho\) is

\[
\begin{array}{c}
N_1 \mapsto \text{node}_1, \ N_2 \mapsto \text{node}_2, \ldots, \ N_{10} \mapsto \text{node}_{10}
\end{array}
\]

(1.150)

(with \(\rho(N_i)\) undefined for \(i > 10\)); while the two ascriptions children and data are as follows (the id ascription is defined in the obvious way):

\[
\begin{align*}
\text{children}(\text{node}_1) & = [\text{node}_2 \ \text{node}_3] \quad \ldots \quad \text{children}(\text{node}_5) = [\text{node}_8] \quad \ldots \quad \text{children}(\text{node}_{10}) = []
\end{align*}
\]

and

\[
\begin{align*}
\text{data}(\text{node}_1) & = \{[1], [2], [5], [8], [1 2], [1 5], \ldots, [8 5 1], \ldots, [1 2 5 8]\} \\
& \vdots \\
\text{data}(\text{node}_9) & = \{[8], [1 8], [8 1], [2 8], \ldots, [5 8 1], [2 5 1 8], \ldots\} \\
& \vdots \\
\text{data}(\text{node}_{10}) & = \{[1 2 5 8]\}.
\end{align*}
\]

Observe the equation for \text{data}(\text{node}_1). At this point we do not know anything about what list appears at \text{node}_1 (a complete lack of knowledge signified by the inscription \(N_1 = ?\)), so the data field of \text{node}_1 is entirely unconstrained: it contains all possible lists of length four obtained by permutations of four objects taken four at a time (\(P(4, 4) = 4! = 24\) total); plus all possible lists of length three obtained by permutations of four objects taken three at a time (\(P(4, 3) = 24\) total); plus all possible lists of length two obtained by permutations of four objects taken two at a time (\(P(4, 2) = 12\), plus all possible lists of length one (4), plus the empty list, for a sum total of \(24 + 24 + 12 + 4 + 1 = 65\) different lists. The data
Figure 1.20: Formal Vivid proof solving the Mergesort puzzle of Section 1.8.1.

ascription maps every “question mark node” (e.g., the nodes labeled by \( N_6 \) or \( N_8 \)) to the same set of 65 lists. Hereafter we will denote this set of 65 lists by \( \mathcal{L} \). By contrast, the data ascription for node 0 (the node labeled by \( N_0 \)) is subject to the constraint that all list values must contain 8, so this narrows down the possibilities to a total of 24 + 18 + 6 + 1 = 49. Further down, the value of data for node 10 is completely determined—the singleton \( \{ [1 \ 2 \ 5 \ 8] \} \). The named system state corresponding to any of the diagrams shown in connection with the Mergesort puzzle is likewise defined. The children ascription and the constant assignment remain the same in every case; while the data value is specified in accordance with the preceding conventions.

Extracting the appropriate system state from a given diagram can be viewed as the task of computing a parsing function \( \phi \) that takes a concrete two-dimensional representation and produces an abstract syntax tree for it. Conversely, reconstructing a diagram from the underlying system state can be seen as computing an “unparsing” function \( \psi \) that proceeds in the reverse direction, rendering system states graphically. As with customary parsing and unparsing, we have

\[
\psi(\phi(d)) = d \quad \text{and} \quad \phi(\psi(\sigma)) = \sigma
\]

for all diagrams \( d \) and system states \( \sigma \), where the first identity is understood to obtain up to topological equivalence.\(^{30}\)

From a practical standpoint, most of the effort required to build a Vivid language would be alloted to the implementation of these two functions. In the case of the Mergesort puzzle, both \( \phi \) and \( \psi \) can be computed efficiently—in low polynomial time—using standard graph-theoretic algorithms.

Regarding diagram parsing, we note that according to the semantics of Vivid, system object identity persists throughout the course of a proof, but given that naming is optional (indeed, figuring out what names get assigned to what objects could be part of the problem to be solved), it is clear that diagrams might underdetermine object identity. That is, one and the same diagram could give rise to different system states depending on how we match up the various iconic elements of the diagram with the underlying system objects. We leave it to the designers of the diagram parser to lay down conventions that eliminate such ambiguities. For instance, in the case of the Mergesort puzzle, it might be decided that the root node is always the first object, followed by its children in left-to-right-order, followed by its grandchildren in the same order, etc. This would keep the identity of the objects fixed even if the nodes were not labeled. In graphical domains where such conventions are not obvious or are too computationally expensive, we can always require that all

\[^{30}\text{These are unnecessarily coarse approximations. We could leverage our knowledge of the domain to further cut down the possibilities drastically. For instance, we know that at the top node only lists of length four could appear—or, in general, only lists of the exact same length as the unique list that appears at the bottom node representing the output. Further, we know that if any node has only lists of \( n \) items as possible values, then the left and right children can respectively only have lists of length \( \lfloor n/2 \rfloor \) and \( \lceil n/2 \rceil \) as possible values, and so on. In this manner cardinality constraints would propagate down the graph and significantly curtail the values of the data ascription. This would be important for an efficient implementation of the Mergesort puzzle, but it is not necessary for our present purposes.}\]

\[^{31}\text{Diagrammatic identity in general can be a vague notion (e.g., when exactly can we say that two drawings depict the same mountain range?), and this is part of the reason why logicians and mathematicians have had a skeptical attitude towards diagrams (Quine’s dictum “No entity without identity” comes to mind). Nevertheless, there are many cases, particularly in discrete domains, where we can formulate rigorous necessary and sufficient conditions for diagram identity.}\]
objects of interest are labeled without compromising the ability to pose problems that ask one to determine which object has what name, since an object can have multiple names (technically, more than one constant symbol can be mapped to the same object), so a heterogeneous proof could reveal, say, that “person p₁” is in fact “Andrew.” But, in general, the details of how diagrams are parsed will vary and cannot be specified in advance.

Finally, Figure [1.20] shows the formal Vivid proof that solves the Mergesort puzzle discussed in Section [1.8.1]. We conclude with a detailed analysis of this proof. Let σ₁, . . . , σ₆ be the system states corresponding to the six diagrams that appear in Figure [1.17] and let D₁, . . . , D₆ be the states corresponding to the diagrams of Figure [1.16] and Figure [1.17] respectively. For any i = 1, . . . , 14, we write τᵢ to denote the named state (σᵢ; ρ), where ρ is the constant assignment (1.150).

Recalling that composition is right-associative, we see that the proof in Figure [1.20] is a sentential proof D, as it is of the form

\[ D = Δ₁; ⋯; Δ₆; D', \]

i.e., a composition of six diagrammatic steps Δ₁, . . . , Δ₆ followed by a sentential deduction D' of the form

\[ \text{cases by } F₁, . . . , Fₖ: \ (σ₁; ρ₁) → D₁ | ⋯ | (σₙ; ρₙ) → Dₙ, \]

a diagrammatic-to-sentential case analysis. The starting point for the proof is the context

\[ γ₁ = (β₁; τ₁), \]

(1.152)

where β₁ contains the five universally quantified clauses of our axiomatization and the aforementioned lemma. This is the context in which the entire proof D will be evaluated.

Let us see why the first step Δ₁, the diagrammatic inference

τ₂ by thinning with union-axiom

succeeds. According to the semantics of thinning (Figure [1.11]), this step will be valid provided that

\[ τ₁ ⊩_{\text{union-axiom}} τ₂, \]

i.e., provided that τ₁ entails τ₂ with respect to union-axiom. This means that every alternative way of extending τ₁ w.r.t. τ₂ must falsify union-axiom (for an arbitrary variable assignment). More precisely, it must be the case that for every named state τ = (σ; ρ) such that Alt(τ₁, τ₂, τ) we have

\[ I(σ; ρ)/χ(\text{union-axiom}) = \text{false} \]

(1.153)

for all χ. Pick any such τ. Since τ₁ and τ₂ share the same constant assignment ρ, the only way τ can be an alternative extension of τ₁ w.r.t. τ₂ is if we have Alt(σ₁, σ₂, σ) (Definition 11). The only state σ that qualifies as such an alternative is the one that is identical to σ₁ except that the data ascription maps node₆ to the set of all lists in L that do not contain 8. It is easy to see that (1.153) holds in that state. Indeed, consider an arbitrary χ. By part (b) of Lemma 13 union-axiom will be false in (σ; ρ) and χ if there are some nodes node₁, node₄, and node₇ such that

\[ I(σ; ρ)/χ[v₁→node₁, v₂→node₄, v₃→node₇](\text{valley}(v₁, v₂, v₃) ∨ \text{peak}(v₁, v₂, v₃) ⇒ \text{union}(v₁, v₂, v₃)) = \text{false}. \]

Let these three nodes be node₉, node₆, and node₇, respectively (i.e., the nodes labeled by N₉, N₆ and N₇). For these nodes we clearly have:

\[ I(σ; ρ)/χ[v₁→node₉, v₂→node₆, v₃→node₇](\text{valley}(v₁, v₂, v₃) ∨ \text{peak}(v₁, v₂, v₃)) = \text{true} \]

(since the nodes form a valley), and yet

\[ I(σ; ρ)/χ[v₁→node₉, v₂→node₆, v₃→node₇](\text{union}(v₁, v₂, v₃)) = \text{false}. \]

(1.154)

(1.154) holds by virtue of Lemma 8 because for every list L in the data field of node₉ in σ and for every list L' in the data field of node₆ in σ and every list L'' in the data field of node₇ in σ, we have

\[ ¬R₆(L, L', L''), \]
the reason being that every such \( L \) contains 8 but no such \( L' \) contains 8 (because \( \text{data}(node_7) \) in \( \sigma \) contains only one list value, \([1]\)) and no such \( L' \) contains 8 (by virtue of \( \sigma \) being an alternative extension of \( \sigma_1 \) w.r.t. \( \sigma_2 \)).

Let us now examine the second step:

\[ \tau_3 \text{ by thinning with halves-axiom.} \]

As with the previous application of thinning, this step is valid only if

\[ \tau_2 \models \text{(halves-axiom)} \tau_3, \]

meaning that any named state \( \tau = (\sigma; \rho) \) that is an alternative extension of \( \tau_2 \) w.r.t. \( \tau_3 \) must falsify halves-axiom. As before, because the constant assignment does not change, the only way we can have \( \text{Alt}(\tau_2, \tau_3, \tau) \) is if we have \( \text{Alt}(\sigma_2, \sigma_3, \sigma) \). And given that in \( \sigma_2 \) the data field of \( node_6 \) contains all and only those lists that contain 8, \( \sigma \) is an alternative extension of \( \sigma_2 \) w.r.t. \( \sigma_3 \) iff it is a list in \( \mathcal{L} \) that contains 8 and has length greater than one, e.g., \([2 \ 5 \ 8]\). But in that state halves-axiom is falsified (with \( node_3, node_6, \) and \( node_7 \) providing the counterexample peak), hence the thinning step is sanctioned. Similar rationales justify the remaining four thinning steps, although the very next step and the last step are more interesting because each cites two formulas as justification for the thinning. In these cases it would seem that evaluation would require an inordinate amount of work, but in fact there are simple heuristics exploiting locality of reference\(^{32}\) that enable such steps to be evaluated efficiently.

We come finally to the case analysis, which turns on the claim that from the state \( \sigma_7 \) and on the basis of the lemma, there are only two possible states, \( \sigma_8 \) and \( \sigma_{12} \). Symbolically,

\[ (\sigma_7; \rho) \models \text{(lemma)} \{(\sigma_8; \rho), (\sigma_{12}; \rho)\}. \tag{1.155} \]

Consulting Definition\(^{12}\) we see that (1.155) holds iff for every \( \langle \sigma'; \rho' \rangle \) such that

\[ \text{Alt}(\sigma_7; \rho), \{(\sigma_8; \rho), (\sigma_{12}; \rho)\}, (\sigma'; \rho') \]

we have \( I_{(\sigma'; \rho')/\chi} \text{(lemma)} = \text{false} \) for all \( \chi \). Again, because the constant assignment does not change, (1.156) holds iff

\[ \text{Alt}(\sigma_7, \{\sigma_8, \sigma_{12}\}, \sigma') \]

(by Definition\(^{11}\)).

Now there are four alternative extensions of \( \sigma_7 \) w.r.t. \( \{\sigma_8, \sigma_{12}\} \), call them \( \sigma_A, \sigma_B, \sigma_C, \) and \( \sigma_D \). They agree on everything except the data values of \( node_4 \) and \( node_5 \). Specifically:

\[
\begin{align*}
\sigma_A(\text{data}, node_4) &= \mathcal{L} \setminus \{\[2\], \[5\]\} \\
\sigma_A(\text{data}, node_5) &= \mathcal{L} \\
\sigma_B(\text{data}, node_4) &= \mathcal{L} \setminus \{\[2\]\} \\
\sigma_B(\text{data}, node_5) &= \mathcal{L} \setminus \{\[5\]\} \\
\sigma_C(\text{data}, node_4) &= \mathcal{L} \setminus \{\[5\]\} \\
\sigma_C(\text{data}, node_5) &= \mathcal{L} \setminus \{\[2\]\} \\
\sigma_D(\text{data}, node_4) &= \mathcal{L} \setminus \{\[2\]\} \\
\sigma_D(\text{data}, node_5) &= \mathcal{L} \setminus \{\[5\]\}
\end{align*}
\]

The relevant parts of these four states can be depicted graphically as follows:

\[\begin{align*}
\sigma_A: & \quad N_2 = ? \\
N_4 &= \mathcal{L} \setminus \{\[2\], \[5\]\} & N_5 &= ? \\
N_8 &= \{2 \ 5\} \\
\sigma_B: & \quad N_2 = ? \\
N_4 &= \mathcal{L} \setminus \{\[5\]\} & N_5 &= \mathcal{L} \setminus \{\[5\]\} \\
N_8 &= \{2 \ 5\} \\
\sigma_C: & \quad N_2 = ? \\
N_4 &= \mathcal{L} \setminus \{\[2\]\} & N_5 &= \mathcal{L} \setminus \{\[2\]\} \\
N_8 &= \{2 \ 5\} \\
\sigma_D: & \quad N_2 = ? \\
N_4 &= ? & N_5 &= \mathcal{L} \setminus \{\[2\], \[5\]\} \\
N_8 &= \{2 \ 5\}
\end{align*}\]

\(^{32}\)Although we do not have sufficient space here to elaborate, the main idea is that if a thinning step has refined the attribute values of certain system objects, then any universally quantified formulas cited as justification for the thinning step are likely to be falsified, in the proper alternative extensions, by those particular objects.
A routine calculation will confirm that in each of these four states the conjunction of union-axiom and halves-axiom is false no matter what list values are chosen for node$_4$, node$_5$, and node$_8$.

1.9 Seating puzzles

We now consider a puzzle that has become somewhat of a classic in discussions of diagrammatic reasoning. Presented (and to the best of our knowledge, devised) by Barwise and Etchemendy (1990), it is typical of the sort of problems found in the analytical section of the GRE, as well as typical of cognitive science experiments investigating spatial reasoning (Byrne and Johnson-Laird 1989)\[33\].

The puzzle is described as follows:

Four people $A$, $B$, $C$, and $D$ are to be seated in a row of five seats. The seating arrangement must satisfy the following three conditions:

1. $A$ and $C$ should flank the empty seat.
2. $C$ should be closer to the middle seat than $B$.
3. $B$ and $D$ should be seated next to each other.

On the basis of this information:

(a) Prove that the empty seat cannot be either in the middle or on either end.
(b) Can it be determined who must be seated in the middle seat?
(c) Can it be determined who is to be seated on the two ends?

Looking ahead, all three questions are answered with one Vivid deduction, shown in Figure [1.21]. Note that the Vivid system we are about to specify is not hardwired to this particular puzzle, but allows for the formulation and solution of a wide variety of seating puzzles (involving arbitrarily many seats, persons, names thereof, and combinations of constraints). Extensions of the system to two-dimensional puzzles involving several rows of seats would not be difficult.

If we make the convention that seats are represented by underscores and that a square box designates an empty seat,

\[
\begin{array}{ccccc}
A & \square & C & B & D \\
\end{array}
\]  

(1.158)

Note that this particular arrangement satisfies all three constraints. We will also make the natural convention of identifying the five chairs from left to right with the five integers $1, \ldots, 5$, so that the leftmost chair is chair 1 and the rightmost is chair 5. An underscore by itself indicates that we do not know who is to be placed in that seat, or more precisely, that any candidate could be seated there. Thus, for instance, the “diagram”

\[
- - A - -
\]  

(1.159)

indicates a seating arrangement in which A is seated in chair 3, but we do not know who is seated in the other chairs.

We now introduce a simple Vivid language that lets us solve problems in this domain. Intuitively, the system objects will be persons, or more abstractly, objects to be seated (including, by convention, square boxes, serving to identify unoccupied seats); and their attribute values will be seat numbers, or more precisely, sets of seat numbers, allowing for incomplete information. Accordingly, a system state will map each such object to a set of positive integers representing chairs. For instance, diagram [1.158] corresponds to the system state

\[
\text{seat}(A) = 1, \text{seat}(\square) = 2, \text{seat}(C) = 3, \text{seat}(B) = 4, \text{seat}(D) = 5,
\]

which is a world, while diagram [1.159] is captured by the system state:

\[
\text{seat}(A) = 3, \text{seat}(\square) = \{1, 2, 4, 5\}, \text{seat}(C) = \{1, 2, 4, 5\}, \text{seat}(B) = \{1, 2, 4, 5\}, \text{seat}(D) = \{1, 2, 4, 5\}.
\]

The vocabulary for this instance of Vivid consists of the binary symbol for identity (=), along with six relation symbols, whose intuitive semantics are as follows:

\[33\] The puzzle is also discussed by Shin (2004), by Barwise and Etchemendy (1996), and others.
cases by flanking$(A, C, \Box)$, distinct-seats:

$$\begin{align*}
| & A \Box C \rightarrow \text{observe goal} \\
| _- A \Box C & \rightarrow \text{begin} \\
& \text{observe } \neg \text{adjacent}(B, D); \\
& \text{absurd } \neg \text{adjacent}(B, D), \neg \text{adjacent}(B, D); \\
& \text{goal by absurdity} \\
| _- _- A \Box C & \rightarrow \text{begin} \\
& \text{observe } \neg \text{closerToCenter}(C, B); \\
& \text{absurd } \neg \text{closerToCenter}(C, B), \neg \text{closerToCenter}(C, B); \\
& \text{goal by absurdity} \\
| C \Box A - & \rightarrow \text{begin} \\
& \text{observe } \neg \text{closerToCenter}(C, B); \\
& \text{absurd } \neg \text{closerToCenter}(C, B), \neg \text{closerToCenter}(C, B); \\
& \text{goal by absurdity} \\
| _- C \Box A & \rightarrow \text{begin} \\
& \text{observe } \neg \text{adjacent}(B, D); \\
& \text{absurd } \neg \text{adjacent}(B, D), \neg \text{adjacent}(B, D); \\
& \text{goal by absurdity} \\
| _- _- C \Box A & \rightarrow \text{observe goal}
\end{align*}$$

Figure 1.21: A Vivid deduction solving the seating puzzle of Barwise and Etchemendy (1990).

1. flanking$(x, y, z) \equiv x$ and $y$ are flanking $z$.
2. adjacent$(x, y) \equiv x$ and $y$ are seated next to each other.
3. atEnd$(x) \equiv x$ is seated at an end seat (seat 1 or 5).
4. middle$(x) \equiv x$ is seated in the middle seat.
5. closerToCenter$(x, y) \equiv x$ is seated closer to the middle seat than $y$.
6. sameSeat$(x, y) \equiv x$ is assigned the same seat as $y$. (This will hold iff $x = y$).

We let $x, y, z, \ldots$, possibly with subscripts, serve as variables. The constant names are $A, B, C, D, E, \ldots$ and $\Box$, also possibly with subscripts. The attribute structure here is parameterized over the number of seats, $n > 1$:

$$A_{S_n} = (\text{seat} : \{1, \ldots, n\}; \{R_1, R_2, R_3, R_4, R_5, R_6\})$$

1. $R_1 \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\} \times \{1, \ldots, n\}$, defined as follows: $R_1(i, j, k) \iff |i - k| = |j - k| = 1$. This interprets flanking via the profile $[(\text{seat}, 1), (\text{seat}, 2), (\text{seat}, 3)]$.
2. $R_2 \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$, defined as $R_2(i, j) \iff |i - j| = 1$. This interprets adjacent via the profile $[(\text{seat}, 1), (\text{seat}, 2)]$.
3. $R_3 \subseteq \{1, \ldots, n\}$, defined as $R_3(i) \iff i = 1 \lor i = n$. This interprets atEnd via the profile $[(\text{seat}, 1)]$.
4. $R_4 \subseteq \{1, \ldots, n\}$, defined as $R_4(i) \iff n$ is even $\land i = n/2$. This interprets middle via the profile $[(\text{seat}, 1)]$.
5. $R_5 \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$, defined as $R_5(i, j) \iff |i - \frac{n+1}{2} < |j - \frac{n+1}{2}|$.

This interprets closerToCenter via the profile $[(\text{seat}, 1), (\text{seat}, 2)]$. 

63
6. $R_6 \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$, defined as $R_5(i, j) \iff i = j$. This interprets sameSeat via the profile

$$[(\text{seat}, 1), (\text{seat}, 2)].$$

The identity symbol is interpreted by the identity relation on the set of objects. An obvious constraint is that distinct objects must be placed in distinct seats. This will be captured by the sentence

$$\forall x, y. \text{sameSeat}(x, y) \Rightarrow x = y.$$ 

For brevity, we will refer to this formula as \textit{distinct-seats}. We will require its presence in every initial assumption base.

Even though diagrams in this system are particularly simple, being one-dimensional and consisting of plain text, a few remarks on how to parse them are in order. This is a domain in which every system object needs to be labeled in order to be uniquely identifiable, because the identity of a system object cannot be tacitly inferred from a diagram. The reason is that an object here does not occupy a fixed place in every diagram across a given proof, in contrast with the Mergesort puzzle or the map-coloring puzzle of the next section, where a system object (a graph node or a map region) has a unique diagrammatic location throughout a proof. Accordingly, in an actual implementation of this system, the user would be required to fix two parameters at the beginning of a session (i.e., prior to entering and evaluating a proof):

- The number of objects to be seated, $k$, cannot exceed the number of seats $n$, although we could have $k < n$, e.g., we could have eight seats and only five objects to seat.
- For instance, in the case of the particular puzzle discussed in the beginning, the user would specify five chairs and the five objects $A, B, C, D$, and $\square$. This would let the implementation know that every system state thereafter would consist of $k$ objects $s_1, \ldots, s_k$, named throughout by the constant assignment $\rho = \{c_1 \mapsto s_1, \ldots, c_k \mapsto s_k\}$.
- Of course this constant assignment could be subsequently extended, if part of the problem involves determining the referents of certain other additional names. For instance, in the opening puzzle, the second question could be expressed by imposing the constraint $\text{middle}(G)$, where the identity of $G$ is initially unknown; the question could then be answered by proving $G = C$. If a system object $s_i$ does not appear in a diagram (or more precisely, if the corresponding name $c_i$ does not), then $\text{seat}(s_i)$ is defined as the set of all seats that are free in that diagram. For instance, $B, C, D$, and $\square$ do not appear in (1.159), and therefore in the system state (1.160) obtained by parsing (1.159), the seat ascription maps the corresponding objects to the set of all chairs that are unoccupied in that diagram, namely, $\{1, 2, 4, 5\}$. With these conventions in place, the implementation can readily parse any given diagram (a sequence of $n$ underscores, each of which is either blank or has exactly one of the $k$ names placed on it, and such that no name appears more than once) into a unique named state.

We now discuss the solution presented in Figure 1.21. That deduction, when evaluated in a context comprising the minimal-information diagram

$$- - - - -$$

and an assumption base that contains the distinct-seats postulate along with the three given constraints (formalized in this language as $\text{flanking}(A, C, \square)$, $\text{closerToCenter}(C, B)$, and $\text{adjacent}(B, D)$), will derive the conclusion

$$\neg \text{atEnd}(\square) \land \neg \text{middle}(\square) \land \text{middle}(C),$$

which we abbreviate as goal. The Vivid deduction is a faithful representation of the informal reasoning that a human problem-solver would use when tackling the puzzle, which is roughly as follows. Knowing that the empty seat must be between $A$ and $C$, we can distinguish three main cases:

1. $A \square C -$
2. $\_ A \square C _$
3. $\_ \_ A \square C$

\[\text{Strictly speaking, we should make the attribute structure automorphic if we are to formalize the realization of the identity relation symbol via a profile, but this is a minor point.}\]

\[\text{For instance, an object might not appear at all in the seating arrangements during the first few steps of a proof, when we do not yet have enough information to place it correctly, but might appear in the diagram subsequently, once such information is deduced.}\]

\[\text{In a preliminary text-based implementation of this Vivid system, we use \textit{X} (possibly followed by other letters and/or digits) to stand for \textit{C}. We can think of these as special “dummy” or “null” objects to be seated, marking unoccupied chairs, although logically speaking we might as well think of them as persons, and indeed it would make no essential difference if we did away with them altogether.}\]
By symmetry, there are three mirror cases, obtained from the above by flipping the positions of \( A \) and \( C \), for a total of six cases, as shown in Figure 1.21. These six cases represent the only possibilities consistent with the flanking constraint which dictates that \( \Box \) must be between \( A \) and \( C \). Moreover, from these six cases, only

\[
A \ \Box \ C \ - \ - \\
\text{(1.161)}
\]

and

\[
-\ - \ C \ \Box \ A \\
\text{(1.162)}
\]

are consistent with the other two constraints. Each of the remaining four cases either makes it impossible for \( B \) and \( D \) to be adjacent or else fails to place \( C \) closer to the middle than \( B \). So in those cases \textit{goal} follows trivially by contradiction, while in cases (1.161) and (1.162) \textit{goal} follows by simple inspection (by applying \textbf{observe}). The derivation of \textit{goal} explicitly answers the first two questions of the puzzle. For the third question, a simple glance at the proof reveals that in the only two cases that do not involve a contradiction (namely, the top and bottom clauses in Figure 1.21 i.e., cases (1.161) and (1.162)), there are no two unique persons who must be seated on the two ends; any one of \( A, B, \) and \( D \) could be placed on an end seat.

As is the case with map-coloring puzzles, a system state extracted from a seating diagram might contain worlds which are intuitively impossible by the nature of the problem. Suppose, e.g., that we have five chairs and the same five objects. Two of them are

\[
w_1 = \{ \text{seat}(A) = 1, \text{seat}(\Box) = 2, \text{seat}(D) = 3, \text{seat}(B) = \{4,5\}, \text{seat}(C) = \{4,5\} \}. \ \\
\text{(1.165)}
\]

Technically speaking, there are four worlds extending \( \sigma \). Two of them are

\[
w_2 = \{ \text{seat}(A) = 1, \text{seat}(\Box) = 2, \text{seat}(D) = 3, \text{seat}(B) = 5, \text{seat}(C) = 5 \}, \ \\
\text{(1.166)}
\]

both of which are impossible (given that we do not allow placing more than one person in one seat), and indeed cannot even be drawn according to our diagrammatic conventions. Worlds such as these tend to be naturally overlooked by reasoners, when in fact they must often be taken into account, even if superficially. For example, suppose that the current system state is \( \sigma \) (corresponding to (1.163), and that the constraint

\[
\neg \text{atEnd}(B) \ \\
\text{(1.167)}
\]

is in the current assumption base (along with \textit{distinct-seats}, which is in every assumption base, and possibly other formulas as well). In this context, one might be inclined to think that the diagram

\[
A \ \Box \ D \ B \ C. \\
\text{(1.168)}
\]

i.e., the world

\[
\sigma' = \{ \text{seat}(A) = 1, \text{seat}(\Box) = 2, \text{seat}(D) = 3, \text{seat}(B) = 4, \text{seat}(C) = 5 \}, \ \\
\text{(1.169)}
\]

\(^{37}\text{This could be explicitly shown by appending the following sentence to the goal conjunction:} \text{atEnd}(B) \lor \text{atEnd}(D) \lor \text{atEnd}(A).\)

The deduction of Figure 1.21 can successfully derive this stronger \textit{goal} without any modification. \(^{38}\text{Strictly speaking, the result of parsing will be a named state comprising an ascription such as} \{\text{seat}(s_1) \mapsto \{1\}, \text{seat}(s_2) \mapsto \{2\}, \text{seat}(s_3) \mapsto \{3\}, \text{seat}(s_4) \mapsto \{4,5\}, \text{seat}(s_5) \mapsto \{4,5\}\} \text{along with the constant assignment} \rho = \{A \mapsto s_1, \Box \mapsto s_2, D \mapsto s_3, B \mapsto s_4, C \mapsto s_5\}, \text{but since in this case system objects are uniquely identifiable by their names, we can abbreviate the result in the form of (1.164) without any ambiguity or loss of information.}\)
follows directly from $\sigma$ by thinning with (1.167) alone, when it does not. To see that it does not, recall that for $\sigma'$ to follow from $\sigma$ by thinning with (1.167), it must be the case that every alternative extension of $\sigma$ w.r.t. $\sigma'$ falsifies (1.167).

There are two such alternative extensions, call them $\sigma_1$ and $\sigma_2$:

- $\sigma_1 = \{\text{seat}(A) = 1, \text{seat}(\square) = 2, \text{seat}(D) = 3, \text{seat}(B) = 5, \text{seat}(C) = \{4, 5\}\}$;
- $\sigma_2 = \{\text{seat}(A) = 1, \text{seat}(\square) = 2, \text{seat}(D) = 3, \text{seat}(B) = \{4, 5\}, \text{seat}(C) = 4\}$.

While $\sigma_1$ does falsify (1.167), $\sigma_2$ does not, because it allows for the world in which both $B$ and $C$ are seated in the fourth seat—in which case $B$ is certainly not at the end. Hence, the truth value of (1.167) in $\sigma_2$ is unknown, instead of the requisite false. However, the conjunction of (1.167) and distinct-seats is falsified by both states, because every world in each state falsifies at least one of these two constraints. Thus the diagram (1.168) does follow from $\sigma$ by thinning both with $\neg \text{atEnd}(B)$ and with distinct-seats. In general, the distinct-seats constraint is necessary for eliminating degenerate seating assignments that place multiple objects in one seat; and that is why it is invoked in tandem with the flanking constraint on the top case analysis of the deduction in Figure 1.21.

### 1.10 Map coloring

As another example consider map coloring, where adjacent countries—or more generally, adjacent regions—in a map must receive different colors. We will refer to this as the “adjacency constraint,” or AC for short. AC can be symbolically formulated as follows:

$$\forall r_1, r_2. \text{adjacent}(r_1, r_2) \Rightarrow \neg \text{sameColor}(r_1, r_2),$$

where $r_1$ and $r_2$ range over regions and adjacent and sameColor have the obvious interpretations. It is well-known that any planar map can be colored in a way that satisfies AC using no more than four colors (Appel and Haken 1989). We will suppose that the four available colors are red, green, blue, and yellow ($R, G, B, Y$). This section will illustrate how rigorous reasoning about such problems can be carried out most naturally with a mixture of diagrammatic and symbolic reasoning in Vivid; it will also demonstrate how the assignment of names can be part of a heterogeneous deduction.

Consider the map of the five regions shown in Figure 1.22. A questionmark inside a region indicates that the color of that region is unknown; it could be any of the four possibilities. A set of colors such as $\{R, G\}$ inside a region indicates that the color of that region must be one of the set’s elements. A single color inside a region has the obvious interpretation; e.g., the central region in Figure 1.22 must be colored blue. A region could optionally be given a name. In the example of Figure 1.22 only two regions have names, $c_1$ and $c_3$. Suppose now that we are given the following two sentential constraints, in addition to the information depicted in the figure:

1. $\neg \text{yellow}(c_3)$; (1.170)
2. $\text{sameColor}(c_1, c_2) \land c_1 \neq c_2$. (1.171)

---

39 We assume for simplicity that adjacent is irreflexive.

40 Provided we do not admit disconnected regions, and that adjacency means sharing a line segment (so that, for instance, sharing a single point on the plane is not sufficient to make two regions adjacent).

41 Although this particular example pertains to map coloring, similar techniques are applicable to many constraint satisfaction problems (Marriott and Stuckey 1999). Vivid is particularly apt for reasoning about such problems.
Figure 1.23: A diagrammatic conclusion entailed by the diagram of Figure 1.22 and constraints (1.170) and (1.171).

The intended meanings are obvious: (1.170) requires that $c_3$ must not be colored yellow (hence its color must be either $R$, or $G$, or $B$); while (1.171) requires that $c_1$ and $c_2$ must be distinct regions assigned the same color. Note that the name $c_2$ does not appear in the diagram; it is part of the problem to figure out which region is denoted by that name. Note further that while (1.170) could be expressed diagrammatically, by writing $\{R, B, G\}$ inside $c_3$, constraint (1.171) cannot be expressed diagrammatically.

What can we infer from the given diagram, formulas (1.170) and (1.171), and AC? First, we can deduce that the colors of the two regions flanking the central blue region must be in the set $\{R, G, Y\}$; they cannot be blue by (AC). Further, the color of $c_3$ must be either red or green; it cannot be yellow on account of (1.170).

We can now perform a two-way case analysis:

1. Suppose first that the color of $c_1$ is red. Then, by the preceding conclusions and AC, we can conclude that $c_3$ must be green, while the color of the leftmost region must be yellow. Accordingly, by (1.171), $c_2$ has to be the uppermost region, since all other regions have distinct colors, and it must be red.

2. By contrast, suppose that $c_1$ is green. Then by similar reasoning we can conclude that $c_3$ must be red; the leftmost region must be yellow; and that $c_2$ must be the uppermost region, and it must be green.

Thus we conclude that, in either case, $c_2$ must be the uppermost region, and it must be either red or green, while the color of the leftmost region must be yellow. The diagram representing our final conclusion and depicting all the information we have extracted is shown in Figure 1.23. Note that this is a diagrammatic conclusion that could not be expressed sententially, since it involves labeling the diagram (by placing $c_2$ in the appropriate region).

The above reasoning can be expressed as a heterogeneous deduction, shown in Figure 1.24 and Figure 1.25, where $\tau_0$ is the initial named state shown in Figure 1.22. (A detailed specification of an attribute structure, vocabulary, and interpretation will be provided shortly, but the depicted proof should be sufficiently clear at this point even without those details.) Each step of the deduction depicts the propagation of one of the constraints. Formally, the Vivid proof that expresses this reasoning is as follows:

$$
\begin{align*}
\tau_1 & \quad \text{by thinning with AC;} \\
\tau_2 & \quad \text{by thinning with ~yellow}(c_3); \\
\text{cases:} & \\
\tau_3 & \quad \text{by thinning with AC;} \\
\tau_4 & \quad \text{by thinning with AC;} \\
\tau_5 & \quad \text{by thinning with AC;} \\
\tau_6 & \quad \text{by thinning with \ sameColor}(c_1, c_2) \land c_1 \neq c_2; \\
\tau_7 & \quad \text{by widening} \\
\tau_8 & \quad \text{by thinning with AC;} \\
\tau_9 & \quad \text{by thinning with AC;} \\
\tau_{10} & \quad \text{by thinning with \ sameColor}(c_1, c_2) \land c_1 \neq c_2; \\
\tau_{11} & \quad \text{by widening} \\
\end{align*}
$$
In the interest of rigor, we close by presenting a detailed specification of a Vivid language for map coloring. Let \( \text{Region} \) be a universe of regions. An appropriate attribute structure for this language is:

\[
\mathcal{A}_R = (\text{id} : \text{Region}, \text{neighbors} : \mathcal{P}_{\text{fin}}(\text{Region}), \text{color} : \{R, B, G, Y\}; \{R_1, R_2, R_3, R_4, R_5, R_6\})
\]

where:

1. \( R_1 \subseteq \text{Region} \times \mathcal{P}_{\text{fin}}(\text{Region}) \), defined as follows: \( R_1(r, S) \Leftrightarrow r \in S \).
2. \( R_2 \subseteq \{R, B, G, Y\} \times \{R, B, G, Y\} \), defined as \( R_2(c_1, c_2) \Leftrightarrow c_1 = c_2 \).
3. \( R_3 \subseteq \{R, B, G, Y\} \) is the “red” property, i.e., \( R_3(c) \Leftrightarrow c = R \). Likewise, \( R_4, R_5, R_6 \) correspond to blue, green, and yellow, respectively.

The vocabulary contains seven relation symbols: A binary \emph{adjacent} symbol, interpreted by \( R_1 \) with profile

\[
[(\text{id}, 1), (\text{neighbors}, 2)];
\]

a binary \emph{sameColor} symbol, interpreted by \( R_2 \) with profile \([(\text{color}, 1), (\text{color}, 2)]\); a binary identity symbol \( = \), interpreted by the corresponding identity relation on the \( \text{id} \) attribute, with profile \([(\text{id}, 1), (\text{id}, 2)]\); and four unary relations, \( \text{red}, \text{blue}, \text{green}, \text{yellow} \), interpreted by \( R_3—R_6 \), respectively, each with profile \([(\text{color}, 1)]\).

### 1.11 Related Work

We have derived much inspiration from the seminal work of Barwise, Etchemendy, and others on Hyperproof (Barwise and Etchemendy 1995b). Chief among the many contributions of Hyperproof were its emphasis on incomplete information and its ability to reason about ambiguous (partially determined) diagrams. These choices are not only pedagogically sound, since there are many types of reasoning problems in which students are given an incomplete sketch and are asked to fill in the gaps by way of inference; but they are also apt design choices for visual reasoning systems in general, as oftentimes the information that agents extract from a perceived image is incomplete, either because parts of the image are visually unclear or because they are not sure how to interpret them.

Important differences between Vivid and Hyperproof include the following:

1. Hyperproof is specifically built for reasoning about simple blocks worlds. Vivid, by contrast, is a domain-independent framework.

2. Hyperproof’s treatment of incomplete information is ad hoc. For instance, although it is possible to signify that the size of a block is unknown, one cannot indicate that it is, say, large or medium but not small. Although these restrictions are due to the limitations of the available palette of visual abstraction tricks, they are reflected in the underlying semantics of the system (Barwise and Etchemendy 1995a, Section 7.5.2). Consequently, if a new abstraction trick is added to the system, the entire semantics would need to change. By contrast, Vivid’s mechanism for handling incomplete diagrammatic information via arbitrary sets of values is entirely general, and its semantics are decoupled from any particular set of abstraction tricks. Note carefully that this does not mean that an implementation of Vivid would not need to use abstraction tricks (it would), but only that, unlike Hyperproof, such tricks are not baked into the underlying semantics.

3. Vivid is based on the key DPL ideas of representing assumption scope with context-free block structure and formalizing the denotation of a proof as a function over assumption bases. These two ideas have several advantages for formalizing Fitch-style natural deduction (Arkoudas 2000, Arkoudas n.d.). The standard Fitch practice—adopted by Hyperproof—of capturing assumption scope by drawing nested vertical lines might be viable for pedagogical purposes but would not scale to realistic proofs any more than drawing vertical lines to represent lexical scope in programming languages (instead of the usual begin-end pairs or curly braces) would scale to realistic programs. But it is not just a question of practicality; a precise abstract syntax goes hand-in-hand with formal semantics, the importance of which we discuss below.
4. Vivid has a formal big-step evaluation semantics in the style of Kahn and Plotkin (Kahn 1987, Plotkin 1981). This is not to say that Hyperproof does not have precise semantics or that its semantics cannot be formally defined; only that it does not draw on the same techniques from programming language theory. We stress that this is not an issue of mere stylistic differences in presentation. Casting a formal semantics in a style such as we have used carries significant advantages, especially in metatheoretic investigations, where many arguments take the form of induction proofs on derivations. In general, such a semantics is an invaluable tool for reasoning about proofs in the system, and for evaluating the correctness of algorithms that manipulate such proofs.

5. Because it is based on DPLs, Vivid is extensible from its present form as a proof-checking framework into a Turing-complete programmable system allowing the user to formulate arbitrary tactics (methods) combining diagrammatic and sentential inference steps, in such a way that the soundness of the methods is guaranteed by the formal semantics of the language (see (Arkoudas 2001b) for an example of how such extensions are actually performed; we are currently working on such an extension for Vivid). In such a framework, heterogeneous proofs such as the solutions to Mergesort puzzles could be discovered automatically. It is not at all clear how Hyperproof could be made programmable, let alone in a way that would guarantee soundness.

6. Hyperproof is proprietary; Vivid is in the public domain.

The work of Konolige and Myers on reasoning with analogical representations (Myers and Konolige 1995) is somewhat similar in spirit to our research, in that it seeks to formulate domain-independent principles for diagrammatic reasoning. However, they do not provide any linguistic abstractions for performing such reasoning. Rather, they outline a set of data structure operations (which they call “the integration calculus”) that can be used to integrate diagrammatic inference into existing reasoning systems, and which can be described as a programming interface. By contrast, we have introduced a precisely defined family of languages for heterogeneous natural deduction, with novel syntax forms and formal semantics. Further, our method for dealing with what they call “structural uncertainty” (incomplete diagrammatic information) is much more general. Finally, our system is strictly more powerful in that it can perform diagrammatic case reasoning; their integration calculus does not have that capability.

DIAMOND (Jamnik 2001) is a system for checking diagrammatic proofs of certain types of arithmetic theorems. The system is designed to reason about natural numbers, and specifically about universally quantified identities of the form \( \forall \ldots , s = t \), where \( s \) and \( t \) are terms built from the numerals 0, 1, 2, \ldots, variables, and operators such as addition, multiplication, etc. A typical example is the identity asserting that the sum of the first \( n \) odd natural numbers is \( n^2 \), symbolically written as

\[
\sum_{i=1}^{n} 2i - 1 = n^2. \tag{1.172}
\]

Diagrammatic proofs are only given for particular instances of the theorem, e.g., for (1.172) one might give a diagrammatic proof for \( n = 4 \), establishing that \( 1 + 3 + 5 + 7 = 4^2 = 16 \). A diagrammatic proof of such a concrete identity is given by representing both terms (\( 1 + 3 + 5 + 7 \) and \( 4^2 \)) as diagrams, and then rewriting both diagrams to a common form. This clearly depends on the system’s ability to represent concrete numeric terms by suitable diagrams. This is possible and indeed intuitive for certain types of terms. E.g., \( 4^2 \) can be represented as a \( 4 \times 4 \) square matrix of dots:

![4x4 Square Matrix Dots](image)

and likewise for any \( n^2 \). It is not so easy for other terms, however, and indeed DIAMOND currently cannot even express some arithmetic theorems.

After the user has successfully carried out several diagrammatic proofs of such concrete instances of the identity in question, the system uses inductive learning techniques in an attempt to automatically extrapolate a schematic proof algorithm capable of taking any number \( n \) and proving the identity for that particular number. If successful, the schematic proof algorithm then needs to be proved correct in a metathetic framework. This is probably the most problematic step of the process, as the problem is undecidable in general. As a result, even though DIAMOND is a proof checker and not a proof finder, it might nevertheless fail to yield a verdict. Therefore, it might make more sense to incorporate abstraction devices into the diagrams in a disciplined way, and attempt from the outset to give diagrammatic proofs of the general form of the theorem, instead of insisting on dealing with concrete diagrams only.
An attempt to extend the ideas of DIAMOND to continuous domains led to Dr. Doodle (Winterstein, Bundy and Gurr 2004), which is an interactive proof assistant for metric space analysis, primarily intended as an educational tool. The inference rules are specified as “redraw rules,” which can be understood as visual analogues of rewrite rules. A more extensive description of the relevant ideas are presented by Winterstein, Bundy, Gurr and Jammik (2002), who also discuss an application of animation for representing and reasoning about quantification. This is an interesting idea (and novel, to the best of our knowledge), although, as the authors acknowledge, it suffers from the drawback that “it is not suited to being printed (e.g., in textbooks or papers), except as cumbersome comic strips where the simplicity of the representation is lost” (Winterstein et al. 2002, section 6). Nevertheless, we do not view this as a serious issue, insofar a system such as Dr. Doodle is supposed to be a computerized tool. Unlike Vivid, the system is not domain-general; its inference rules are restricted to real analysis. We are also not aware of any formal analysis of this system (proofs of soundness or other results about it), or any discussion of computational complexity issues. It is not clear what proof system (Fitch, sequent-based, etc.) is used for the non-diagrammatic part of the system.

GROVER (Barker-Plummer and Bailin 1992) is a theorem-proving system that uses diagrams to guide the proof search. The system consists of a conventional (sentential) automated theorem prover (ATP), &, augmented with a diagram processor. The diagram processor examines the given diagrams and, based on the extracted information, it constructs an appropriate proof strategy for &. The system has reportedly been used to obtain automatic proofs for the diamond lemma, as well as for the Schröder-Bernstein theorem of ZF. Both are non-trivial results; the Schröder-Bernstein theorem, in particular, has a quite sophisticated sentential proof that is far from even the current state-of-the-art in ATP technology. According to the authors of GROVER, a diagram represents a trail of the objects that are involved in the proof, along with key properties of and relations among such objects. That is an interesting view of diagrams, but it differs markedly from the sense in which they are used in Vivid, where they are essentially treated as visual premises and inference rules are applied to them in the usual step-by-step fashion.

Anderson and McCartney (Anderson and McCartney 2003) present IDR, a system for representing and computing with arbitrary diagrams. A diagram is viewed as a tessellation of a finite two-dimensional planar area, with each tile having a unique triple of numbers $i, j, k$ associated with it, indicating a value in the CMY (Cyan, Magenta, Yellow) color scale. Apart from the spatial relationships between the tiles, the meaning of a diagram is captured mainly via tile coloring, with different colors (or shades of gray) representing different types of information. They introduce a set of operations on diagrams, each of which takes a number of input diagrams of the same dimension and tessellation, and produces a new diagram in which the color value of a tile is a function of the color values of the corresponding tiles of the input diagrams. Among other applications, IDR has been used to solve the $n$-queens problem diagrammatically, to induce correct fingerings for guitar chords, and to answer queries concerning cartograms of the USA. The system is more concerned with diagram computation than with inference; there are no general notions of entailment, soundness, etc. IDR is also not heterogeneous. It is exclusively diagrammatic, in that all the available operations are applied to diagrams, not to combinations of diagrams and sentences.

Barwise and Etchemendy (1990) set out to “do some of the homework needed to develop a general theory of heterogeneous inference” (1990, p. 33). They provide an information-theoretic analysis of heterogeneous inference that draws heavily from situation theory (Barwise and Perry 1983). The key idea underpinning their work is a class of mathematical structures known as infon algebras, which turn out to be Heyting algebras (complete distributive lattices). Their analysis is thoroughly abstract, in that it is couched independently of any particular representational system and any set of syntactic constructs. Our perspective, by contrast, is that of computer science, and particularly that of artificial intelligence and programming language theory and implementation. Our chief objective is not to study heterogeneous inference from a purely mathematical perspective, but to actually design and build a formal framework that mechanizes such inference. That means coming up with an abstract syntax and with formal operational semantics, proving various results about them, and demonstrating the utility of the language via examples. Particular attention has been paid to computational efficiency concerns and to achieving a modular design. We have presented detailed algorithms on how to implement Vivid systems, and heuristics that facilitate the efficiency of such implementations. These issues do not surface at all in the work of Barwise and Etchemendy (1990), simply because it was not the objective of that work to address them. The difference is perhaps best elucidated by the following remark of Barwise and Etchemendy (1990, p. 68) on their analysis of the seating puzzle:

We emphasize that we are presenting a mathematical model that shows the reasoning given above [solving the seating puzzle] to be valid. This is a distinct enterprise from modeling the reasoning itself [our italics]. It is analogous to a model-theoretic proof of the soundness of principles used in a piece of syntactic reasoning. Needless to say, this is not something that the reasoner does in the course of the reasoning.
Our focus, by contrast, has been precisely the modeling of the reasoning itself—the formal modeling of what “the reasoner does in the course of the reasoning.” We want reasoners to be able to communicate certain pieces of heterogeneous reasoning to the machine succinctly and perspicuously, seamlessly integrating diagrams and sentences, in such a way that the formal machine-readable object mirrors the informal reasoning to the greatest possible extent; and we want the machine to automatically check the soundness of the reasoning. This has been our primary concern in this paper (the automatic discovery of such heterogeneous proofs will be our main next goal), and we believe we have achieved it, insofar, for instance, the formal Vivid proof shown in Figure 1.21 does mirror the informal reasoning quite closely, it has a perfectly rigorous syntax and semantics, and it is machine-readable and machine-checkable. The same points distinguish our work from that of Vickers (1989).

Apart from this major difference in objectives, our work diverges from that of Barwise and Etchemendy (1990) in other important respects. Most notably, it is not clear to us that there is a general theory of heterogeneous inference to be had. There are significant theoretical differences between diagrammatic reasoning and sentential reasoning, stemming from fundamental representational differences between diagrams and sentences, a fact which calls into question the existence of a single algebraic or axiomatic characterization that captures both equally well. This is not to say that a single theoretical framework combining both cannot be developed; only that in practice such a framework will have to make an explicit distinction between the two, as we have in the present work, instead of dealing only with infons and keeping completely neutral about representation. A specific example of the type of “fundamental representational difference” we have in mind concerns the inability of diagrams to convey inconsistent information (see page 9), and the lack of a negation operator on diagrams. These points have been recognized in the past, both by philosophers (Sober 1976) and by artificial intelligence researchers (Etherington, Borgida, Brachman and Kautz 1989). In Vivid they are reflected in the fact that system states only have joins, but not meets; they form a union semi-lattice, but not a lattice. The non-existence of a bottom (or “null,” or “inconsistent”) diagram in Vivid has the important benefit of averting the Bar-Hillel–Carnap paradox (Floridi 2004), abbreviated BCP, whereby an inconsistent state of affairs conveys the maximum possible amount of information. In Vivid, a measure of the information content of a diagram (state) $\sigma$ is quite simple: it is the number of worlds excluded by $\sigma$, or, in the language-theoretic view, the number of strings in the complement of $\sigma$.

So a world conveys the maximal amount of information—nothing can express more information. This simple and rather pleasant state of affairs ultimately stems from the aforementioned expressive limitations of diagrams. Given that infon algebras are always lattices, however, and indeed that they always come equipped both with a greatest and with a least element 1 and 0 (Barwise and Etchemendy 1990, p. 39), it is difficult to see how any sensible formulation of an information-content measure in the framework of Barwise and Etchemendy (1990) would not fall prey to BCP.

Swoboda and Allwein (2002) propose a general framework for modeling heterogeneous systems. The examples they present (specifically, the seating arrangement and the age charts) are readily modeled in Vivid, but a meaningful comparison is difficult, as it is not clear what proofs would look like in that framework (the paper does not present any heterogeneous deductions as examples), how soundness would be ensured (no theorems are proved about the framework), or how mechanization would be enabled.

Spider diagrams (Howse, Molina, Taylor, Kent and Gil 2001, Stapleton, Howse, Taylor and Thompson 2004) are an extension of Euler circles and Venn diagrams, used to express constraints on sets. They arose from work on constraint diagrams (Kent 1997), which were introduced as a mechanism for the visual depiction of constraints expressing invariants in object-oriented models specified in UML (Rumbaugh et al. 1999). They combine elements of Euler circles and Venn diagrams. In particular, they relax the requirement of Venn diagrams that all curves must intersect, which retains the intuitive appeal of Euler circles, while allowing for shading, which makes Euler circles more expressive. In addition, 

---

42 Properly normalized. Specifically, the information content of a Vivid state $\sigma$ can be defined as

$$C(\sigma) = \frac{|\{w \mid w \not\in \sigma\}|}{|\{w \mid \exists \sigma, w \not\in \sigma\}|},$$

i.e., as the ratio of the number of all the worlds which do not extend $\sigma$ (and are thus inconsistent with it) over the total number of possible worlds. Accordingly, the state which maps every attribute of every object to the set of all possible attribute values has zero information content (since it excludes no possibilities whatsoever), whereas individual worlds have maximum information content, though never quite equal to 1. If we regard the set of all worlds as the event space of a probabilistic experiment, where every world has the same a priori probability, then we can define the probability $P(\sigma)$ of a state as the ratio of the number of worlds in it over the total possible number of worlds. Accordingly, worlds have the smallest possible (but non-zero) probability, while the all-inclusive state has probability one, and we thus have the classic identity of Bar-Hillel and Carnap (1952):

$$C(\sigma) = 1 - P(\sigma).$$

That is, the information content of a state is inversely proportional to its probability—the central tenet of Shannon’s mathematical theory of communication.

43 Again, assuming that all worlds are equally probable.
they introduce “spiders,” which are generalizations of Peirce’s X-sequences (linked points, which are interpreted disjunctively as asserting that at least one of the points lies in a non-empty region). This work differs from Vivid in several respects. First, spider diagrams are a purely diagrammatic reasoning system, whereas Vivid is a heterogeneous system that integrates sentential and diagrammatic inference (a typical inference rule in Vivid, such as thinning, involves both diagrams and sentences). Second, spider diagrams, being based on Euler and Venn diagrams, are used to depict sets and relationships among sets, so their graphical elements are restricted to closed plane curves and spiders. That is, of course, quite appropriate for object-oriented design, where the main entities of interest are classes, which are naturally interpreted as sets. But there are many cases where we wish to draw and reason about other types of diagrams, say, chess boards, where the objects of interest are chess pieces rather than sets; or graphs, where the objects of interest are not sets but individual nodes and the topological relationships indicate graph-theoretic—insead of set-theoretic—relationships; or blocks worlds, where the spatial relationships of interest are on, above, and below, and potentially left-of and right-of, instead of enclosure, overlap, and separation. Such diagrams can be parsed into named system states, and once an appropriate vocabulary has been chosen, Vivid can be used to carry out heterogeneous reasoning with them. Finally, unlike spider diagrams, the semantics of Vivid are built on a three-valued logic specifically designed to deal with incomplete information.

Wang, Lee and Zeevat (1995) attempt a formal analysis of the general properties of diagrams, in the setting of multi-sorted first-order logic. They do not present a specific axiomatic characterization of diagrams, but give a few suggestive examples of such axioms, e.g., that the “overlaps” relation is symmetric, or that if a point \( x \) is inside a circle \( y \), then \( x \) is not outside \( y \). The authors acknowledge that, technically speaking, what they call a “graphical signature” is simply a multi-sorted signature, and what they call a “graphical theory” is simply a multi-sorted first-order theory, but they nevertheless single out some formal properties which, they argue, achieve a partial characterization of graphical theories, i.e., theories expressing diagrams. These three properties are atomicity, consistency, and maximality. Atomicity means that the only properties that a diagram expresses are “basic facts,” expressed by literals—atomic sentences or negations thereof. Thus, for instance, a theory that purports to capture a diagram does not contain any quantified or any conditional sentences. We agree with the general spirit of this condition, and indeed Vivid does not even use negations of atomic sentences in diagrams (system states). But there is one significant difference, namely, that Vivid allows for disjunctive information in order to accommodate incomplete diagrammatic information and visual ambiguity. Consistency simply “reflects the fact that diagrams cannot convey contradictory graphical information” (1995, p. 353). This important point also applies to Vivid diagrams, as discussed previously. Finally, maximality means that “for a basic property represented by an atomic sentence \( P \) about some objects in a diagram, either \( P \) or \( \neg P \) can be seen in the diagram” (p. 355). That is not the case for Vivid diagrams, again due to the possibility of incomplete information. In Vivid, a “basic property” about “some objects in a diagram” might well have an unknown truth value. This also distinguishes our framework from the work of Etherington et al. (1989) on “vivid knowledge bases.” We believe that the suitability of a three-valued logic for diagrams is amply demonstrated by the examples we have presented, and by systems such as Hyperproof (Barwise and Etchemendy 1995b). As the underlying framework of Wang et al. (1995) is that of classical first-order logic, they do not present any new inference rules, syntax forms, or semantics specifically designed for heterogeneous inference. There are no algorithms presented in the paper, and few detailed remarks on how the proposed approach would be used to mechanize diagrammatic reasoning.

### 1.12 Conclusions

Sentential reasoning has been studied to an astonishing depth over the last century, and by now we have amassed a tremendous amount of knowledge about it, both theoretical and practical. A great number of symbolic-reasoning tools such as theorem provers, proof checkers, model finders, model checkers, planners, logic programming languages for deduction, induction, and abduction, etc., are used on a daily basis to perform all kinds of tasks, from machine learning to exploring mathematics, verifying hardware and software, finding plans for autonomous agents, performing scene understanding for robots, and more.

The progress has been nowhere near as impressive for diagrammatic reasoning, even though AI researchers have long recognized the advantages of analogical representations, and scientists have always used diagrams in problem solving. Indeed, as was pointed out in the introduction, visual modes of presenting information are overwhelmingly more popular (some would say trendier) than sentential modes, both in academia and in business. But in logic and mathematics, and particularly when it comes to formal proofs, the use of diagrams has been meager. This is partly because diagrams acquired a suspect reputation after the advent of the logicist era ushered in by Frege and Russell,
which resulted in an almost exclusive focus on sentential formalisms and stigmatized images as inherently non-rigorous and prone to mistakes. Attitudes towards diagrams have been improving over the last fifteen years (largely as a result of the seminal work of Barwise and Etchemendy), but the stigma has not quite disappeared. There are contemporary reports of mathematicians who go so far as “actually hiding the diagrams and visual arguments in presenting their lectures and proofs” (Wheatley 1997, p. 281).

Our work is another step towards rectifying this discrepancy and putting diagrammatic reasoning on a solid footing by integrating it with symbolic reasoning in a heterogeneous framework. Our results demonstrate that it is possible to perform perfectly rigorous reasoning with and about arbitrary diagrams in combination with regular symbolic reasoning. The last dozen years or so have witnessed an encouraging surge of research activity on diagrammatic reasoning, and similar demonstrations have been carried out in certain particular diagrammatic domains such as Venn diagrams and Peirce diagrams (Shin 1995, Hammer 1995). Our work has broader applicability, as it introduces domain-independent idioms for heterogeneous reasoning and general notions of entailment and soundness for such reasoning.

Specifically, we have provided a rigorous analysis of the computational aspects of heterogeneous reasoning by introducing and studying Vivid, a family of denotational proof languages (DPLs) that combine diagrammatic and symbolic inference in a mechanizable Fitch-style natural-deduction framework. Vivid is based on the notion of attribute systems, and on the use of a three-valued logic to interpret first-order signatures into attribute structures. The design of Vivid enables highly modular implementations by enforcing a sharp separation between the purely graphical task of diagram parsing on one hand, and the system’s syntax, semantics, and underlying diagrammatic inference procedures on the other. The latter are fixed once and for all and proven sound. To obtain a particular instance of Vivid, we need only specify a class of diagrams and an associated signature, interpret the signature via an appropriate attribute structure, and provide a diagram parser. This is somewhat analogous to the way in which the CLP scheme (Jaffar and Lassez 1987) defines a family of constraint logic programming languages, with a specific member of the family obtained by fixing a constraint domain and a corresponding constraint solver and simplifier.

Incomplete information plays a key role in our system, as diagrammatic reasoning in Vivid proceeds largely by the gradual elimination of uncertainty. If a diagram is completely determined, i.e., if it is a world, then it already conveys a maximal amount of information. The combination of diagrammatic and symbolic reasoning is crucial in the process of deriving new information. A typical inference step refines a diagram by using some sententially expressed knowledge to rule out certain possibilities.

We have not discussed how diagrams would be concretely represented within the proof text. That is an important implementation issue, but it concerns the interface of Vivid, not its abstract syntax or semantics. One possibility would be to give names to diagrams and then have those names appear in the proof text, but with hyperlinks. If a user clicks on such a link, a picture depicting the corresponding diagram would appear and the user could view or edit the diagram as necessary, save it as another diagram, etc. Of course, as we have already stressed, how diagrams are actually drawn depends on the domain at hand and is kept separate from other aspects of the language. While this separation of concerns results in a very modular design, one might be concerned that ignoring diagram parsing amounts to brushing off potentially difficult problems that could arise in connection with diagrammatic reasoning. In response, it should be noted that diagram parsing is not quite the same as diagram reasoning, in the same way that parsing formulas and reasoning with them are two different tasks. Reasoning is not concerned with how to identify the structure of a diagram or a formula, but with what to do with that structure. Just as the inference rules of sentential logic operate on the abstract syntax of formulas and reasoning starts after parsing has been completed, so it is with diagrams. Moreover, in very many domains diagram parsing could present challenges. But even if a framework could not handle such domains and was instead limited to working with “simple” diagrams (and Vivid has no such limitations), it could still be an exceedingly useful tool. As we pointed out in the introduction, it is not the visual complexity of diagrams that makes them useful; diagrammatic simplicity and structure are often advantages, not disadvantages. By not being fettered to any particular diagrammatic domain and any particular set of graphical conventions, Vivid gives users the freedom to exercise their creative imagination and invent their own simple diagrams and abstraction tricks, appropriate for their particular applications, and then use the deductive machinery of the language to represent and solve problems in ways that would not be possible in a sentential framework.

Two additional points should be emphasized with regard to this last remark. First, a Vivid proof in an actual computer implementation of the language would not contain sentential descriptions of named system states; it would contain the diagrams themselves, exactly as they appear, for instance, in a picture such as that of Figure 12. As we mentioned above, named states would appear in the proof text essentially as hyperlinks. Clicking on such a link would launch a diagram editor, which would display the diagram and also allow the user to edit the diagram and save the changes, if

73
so desired. This interface is very similar to that of Hyperproof. The user sees and manipulates pictorial diagrams, not symbolic descriptions of system states. Of course, under the hood, Vivid will parse the diagrams into system states and will apply the inference rules to those, in accordance with the theoretical framework which we have developed. Hyperproof does something similar, and indeed any other mechanized system would have to do something similar. But users are shielded from all that—what they see and manipulate are the diagrams themselves. Diagrams, therefore, do not serve as mere annotations to Vivid proofs, but instead play an essential causal role in their dynamic behavior. Second, Vivid does not recast diagrammatic information into a standard formalism such as multi-sorted first-order logic and then resort to the usual deductive mechanisms of that formalism for reasoning (which is what would take place in a framework such as proposed by Wang et al. (1995)). That approach would be disastrous from the perspective of the user (for instance, any sentential solution to the seating puzzle in first-order or higher-order logic would be much less natural than the heterogeneous solution shown in Figure 1.21). Instead, Vivid represents diagrammatic information by formal structures (named system states) that are tailor-made for (potentially indeterminate) diagrams, and then deploys novel inference rules operating on those structures that make heterogeneous inference much more natural—all while letting the user graphically manipulate real diagrams.

Finally, as it stands, an implementation of Vivid would be a type-\(\alpha\) DPL, i.e., a proof checker: it would accept a heterogeneous proof and would either pronounce it sound or else point out a reasoning error. Introducing unrestricted naming and computation will extend Vivid into a type-\(\omega\) DPL framework (Arkoudas 2001b, Arvizo n.d.), capable not only of proof checking but of proof search as well. It would be interesting to see what types of methods can be written in such a setting for the purpose of automating heterogeneous inference, and how they fare in comparison to purely symbolic methods. In addition, such an extension will allow for purely diagrammatic reasoning via pattern matching on diagrams (i.e., essentially diagrammatic rewrite rules), similar to the deductive pattern matching of type-\(\omega\) DPLs such as Athena (Arkoudas n.d.), and for diagrammatic generalization, which are issues that Vivid in its present form does not address.
Figure 1.24: Diagrammatic deduction showing the solution to a map-coloring problem, part 1.
Figure 1.25: Diagrammatic deduction showing the solution to a map-coloring problem, part 2.
Bibliography


