Combining diagrammatic and symbolic reasoning

Konstantine Arkoudas

October 4, 2005

Abstract

We introduce a domain-independent framework for heterogeneous natural deduction that combines diagrammatic and sentential reasoning. The framework is presented in the form of a family of denotational proof languages (DPLs). Diagrams are represented as possibly partial descriptions of finite system states. This allows us to deal with incomplete information, which we formalize by admitting sets as attribute values. We introduce a notion of attribute interpretations that enables us to interpret first-order signatures into such system states, and develop a formal semantic framework based on Kleene's strong three-valued logic. We extend the assumption-base semantics of DPLs to accodomodate 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. In addition, we specify detailed algorithms for implementing proof checkers for the resulting languages, and discuss associated efficiency issues.

1.1 Introduction

Diagrams have been recognized as valuable representational and reasoning tools at least since the days of Euclid. Their utility 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 (Sloman 1971), or *homomorphic representations* in the terminology of Barwise and Etchemendy (Barwise and Etchemendy 1995a). In more plain terms, 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 Charles Peirce observed that a diagram is "naturally analogous to the thing represented" (Peirce 1960).

Consider, for instance, the task of descibing 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 excessively long and complicated, and hence not particularly illuminating. [1] 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 5-year-old. A digital picture will be even more accurate.[2] So, as Hammer observes (Hammer 1995), 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 (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. As another example, consider a map of a lake and try to imagine a sentential description of it. Stenning and Lemon (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.

[1] 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 homomorphic to the corresponding shapes than other sentential descriptions.

[2] In the limiting case, of course, 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.

Nevertheless, it is important to keep in mind that two-dimensionality by itself is neither a necessary nor a sufficient condition for being a diagram. For instance, as Hammer (Hammer 1995) points out, a representation of a picture by a two-dimensional array of numbers encoded under some encryption scheme does not classify 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 very well construct intuitive one-dimensional diagrams. E.g., the following string expresses the fact that the stretch of road between Park Avenue/35th Street and Park Avenue/36th is two-way, whereas that between Park Avenue/36th and Park Avenue/37th is one-way and proceeds from right to left:

```
Park/35th <==> Park/36th <== Park/37th
```

Owing to their representational power, diagrams are extensively used 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 set theory, function graphs in calculus and analysis, planar figures in geometry, bar-, chart- and pie-graphs in economics, circuit, state and 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, Mendeizon and Ullman 1982), semantic networks in AI, graphical user interfaces (GUIs) such as Xerox Parc's "Magic Lenses" (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.

Diagrams are not without drawbacks. While they often excel in depicting particular, concrete objects or situations, they are usually not as good for describing general, abstract structures and relationships. Roughly, the smaller and more concrete the class of models captured by a diagram, the more successful the diagram is likely to be. Spatial constraints tend to pull diagrams toward over-specificity, and end up limiting their generality and expressiveness as a result. To take an extreme example, diagrams cannot express tautological or contradictory information. [3]

Expressive limitations can lead to incorrect inferences. It is known that Euler circles (Euler 1768), for instance, are unsound. This follows from Helly's theorem in convex topology (Eggleston 1969). A simple illustration of the problem, due to Lemon and Pratt (Lemon and Pratt 1997), is the following: consider four sets A, B, C, and D, any three of which have non-empty intersections:

$$A \cap B \cap C \neq \emptyset;$$
$$B \cap C \cap D \neq \emptyset;$$
$$A \cap C \cap D \neq \emptyset.$$

These are three perfectly consistent premises. But any Euler diagram that tried to depict them graphically 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. This is a consequence of a special case of Helly's theorem, which states that if any three out of four convex regions have a non-empty intersection then all four must have a non-empty intersection. Similar negative results hold for other diagrammatic ways of depicting sets and relationships between them, such as Englebretsen's linear diagrams (Englebretsen 1992); see Lemon's article (Lemon 2002) for a thorough discussion.

[3]Pierce diagrams can be viewed as a counterexample, but those rely on so many ad hoc conventions that they cannot be said to be analogical representations.

The complexity of diagrammatic reasoning is another issue. Roughly, there are two types of diagrammatic inference. In one of them, exemplified by Euler circles and Venn diagrams, inference is carried out by drawing appropriate diagrams and then reading 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." [4] In the second type of diagrammatic inference, exemplified in systems such as Hyperproof and in our own VIVID, inference is carried out in a more traditional sense, by deriving new diagrams from diagrams that are given as "premises," or by extracting sentential information from given diagrams. Computational complexity issues have been rigorously investigated for the former, but not for the latter. E.g., for the former, 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, and hence on the first type of diagrammatic reasoning. For instance, it has been shown 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 the present work, it will be seen that even though $\mathcal{NDL}$ proofs (Arkoudas n.d.a) can be checked for soundness in $O(n \log n)$ time in the worst case (where n is the size of the proof), checking VIVID proofs can take exponential time, although it should be noted that in our case most of the complexity derives from dealing with unknown (incomplete) information. It would appear, therefore, that visual inference, at least in some cases, can be significantly more expensive than corresponding sentential reasoning.[5]

For these and other reasons, researchers have concluded that logical reasoning frameworks must be *heterogenerous* 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 correctly observe (Barwise and Etchemendy 1995a), 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. In the example of Barwise and Etchemendy, 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 or about diagrams does not depend on how diagrams are drawn or even on what they mean. In this paper we identify what is common in a great variety of instances of diagrammatic reasoning, and proceed to factor it out and extrapolate it into general principles. In the resulting framework, the type of diagrams used may vary from application to application, but the principles by which we reason with and about diagrams remain the same. This is not unlike other separations that are familiar from traditional, sentential logic: our vocabulary might vary from application to application (we have different constant, relation, and function symbols as dictated by the problem domain), and the interpetation 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.

[4] 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 done, 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. This is related to the notion of *free rides* (Shimojima 1996) in diagrammatic reasoning.

[5] There are alternative viewpoints, however. AI researchers have put forth the notion of *vivid knowledge bases* (Etherington, Borgida, Brachman and Kautz 1989, Levesque 1989), in which deductive retrieval can be performed particularly efficiently. Such knowledge bases consist only of ground sentences, ground inequalities, and universal quantifications. Etherington et al. (Etherington et al. 1989) claim that "the notion of vivid representations ... corresponds well to the kind of information expressed in pictures" and that "much of the information we gain (i.e., perceptually) occurs naturally in vivid form." Likewise, Levesque (Levesque 1989) states that "perhaps the main source of vividly represented knowledge is pictorial information." If that is indeed the case, one would expect pictorial reasoning to be efficient. Lemon (Lemon 2002), however, argues that such claims fail to take into account the type of spatial constraints that limit the expressiveness of diagrammatic representations.

1.2 Notation

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

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

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

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

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

The empty list [] is a *sublist* of every list; no non-empty list is a sublist of []; while a list of the form $L = [x_1\ x_2 \cdots x_n]$ is a sublist of a list of the form $L' = [y_1\ y_2 \cdots y_m]$ iff (1) $x_1 = y_1$ and $[x_2 \cdots x_n]$ is a sublist of $[y_2 \cdots y_m]$; or (2) $x_1 \neq y_1$ and L is a sublist of $[y_2 \cdots y_m]$.

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

$$\overbrace{A \times \cdots \times A}^{n},$$

i.e., the set of all lists of length n with elements drawn from A.[6] Given a (partial) function $f : A \to B$ and elements $x \in A$, $y \in B$, $f[x \mapsto y]$ denotes that function from A to B which maps x to y and agrees with f on every other $x' \in A$. More precisely:

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

For $A' \subset A$, $f \restriction A'$ denotes the restriction of F on A', i.e.,

$$f \restriction A' = \{(x; y) \mid f(x) = y \text{ and } x \in A'\}.$$

Finally, for an arbitrary relation $R \subseteq A_1 \times \cdots \times A_n$, $D(R)$ denotes the set $\{A_1, \dots, A_n\}$.

1.3 Attribute structures and systems

Definition 1: An **attribute structure** is a pair $\mathcal{A} = (\{A_1, \dots, A_k\}; \mathcal{R})$ consisting of a finite collection of sets $A_1, \dots, A_k$ called **attributes**; and a countable collection $\mathcal{R}$ of computable relations, with $D(R) \subseteq \{A_1, \dots, A_k\}$ 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 tacitly assume that $\mathcal{R}$ includes the identity relation on each attribute A_i: $\{(a; a) \mid a \in A_i\}$.

We 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. Further, when the relations of $\mathcal{A}$ are immaterial, we identify $\mathcal{A}$ with its attributes. We can then write $\mathcal{A}$ simply as $l_1 : A_1, \dots, l_k : A_k$, where l_i is the label of A_i. The number of attributes k is the *cardinality* of $\mathcal{A}$, denoted by $|\mathcal{A}|$. We say that $\mathcal{A}$ is *finite* iff every attribute of $\mathcal{A}$ is finite.

[6] With $A^1 = A$.

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 (or all) of the objects $s_1, \ldots, s_n$. If that is the case, $\mathcal{S}$ 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."

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

$$(\{c\}; \textit{hours} : \{0, \ldots, 23\}, \textit{minutes} : \{0, \ldots, 59\}).$$

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:

$$(\{c_1, c_2\}; \textit{hours} : \{0, \ldots, 23\}, \textit{minutes} : \{0, \ldots, 59\}).$$ ■

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:

$$(\{n_1, n_2, n_3\}; \textit{data} : \textit{Bool}, \textit{next} : \{n_1, n_2, n_3, \textit{null}\}),$$

where $\textit{Bool} = \{\mathbf{t}, \mathbf{f}\}$ 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:

$$(\{A, B, C\}; \textit{pos} : \{A, B, C, \textit{floor}\});$$

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):

$$(\{b_1, b_2, b_3, b_4\}; \textit{loc} : \{1, \ldots, 8\}^2, \textit{size} : \{\textit{small}, \textit{medium}, \textit{large}\}, \textit{shape} : \{\textit{cube}, \textit{tet}, \textit{dodec}\})$$ ■

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

$$\delta_i : \{s_1, \ldots, s_n\} \to \mathcal{P}_\infty(A_i) \setminus \emptyset.$$

We refer to each δ_i as the state's **ascription** into A_i. An ascription δ_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. ■

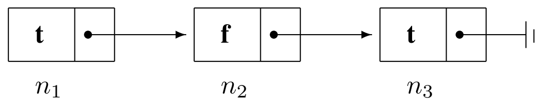


Figure 1.1: A linked list world.

A system that is based on a finite attribute structure has

$$\prod_{i=1}^{k} 2^{(|A_i|-1)^n} = 2^{\left(\sum_{i=1}^{k}(|A_i|-1)^n\right)}$$

states, where n is the number of objects and k the number of attributes. [7] To simplify notation, when δ is a valuation that maps an object s to a singleton $\{a\}$, we write $\delta(s) = a$ instead of $\delta(s) = \{a\}$. 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 δ_i, the state's (unique) ascription into A_i; the context will always make our intentions clear. As an additional convention, given a state σ 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 δ_i for the object s_j.

Our notion of systems and states is similar to the corresponding notions in the model checking field (Clarke, Grumberg and Peled 1999, Bérard et al. 2001), where a system is represented by a collection of variables and a state of a system is modeled by an assignment of a value (drawn from an appropriate domain) to each variable.

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

$$(\{c\}; \mathit{hours} : \{0, \ldots, 23\}, \mathit{minutes} : \{0, \ldots, 59\}).$$

A state σ_1 of this system is given by the following two valuations:

$$\sigma_1 : \quad \mathit{hours}(c) = 15, \mathit{minutes}(c) = 47,$$

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:

$$\sigma_1(\mathit{hours}, c) = 15, \sigma_1(\mathit{minutes}, c) = 47.$$

Suppose we know that it is between 2:30 and 3 past midnight, but do not know exactly how many minutes past 2:30 it is. This state of knowledge can be captured by the following state:

$$\sigma_2 : \quad \mathit{hours}(c) = 2, \mathit{minutes}(c) = \{31, \ldots, 59\}.$$

This state can also be expressed by writing

$$\sigma_2(\mathit{hours}, c) = 2, \sigma_2(\mathit{minutes}, c) = \{31, \ldots, 59\}.$$

Complete lack of information about the time is represented by the state:

$$\mathit{hours} = \{0, \ldots, 23\}, \mathit{minutes}(c) = \{0, \ldots, 59\}.$$

[7] Our term "system state" corresponds roughly to what Barwise et al. (Barwise and Etchemendy 1995b) refer to as "situation." Our notion is much more general, as will be seen.

Example 6: Consider the linked-list system of Example 2. The state

$$\mathit{data}(n_1) = \mathbf{t}, \mathit{data}(n_2) = \mathbf{f}, \mathit{data}(n_3) = \mathbf{t}, \mathit{next}(n_1) = n_2, \mathit{next}(n_2) = n_3, \mathit{next}(n_3) = \mathit{null}$$

depicts the world shown in Figure 1.1. The state

$$\mathit{data}(n_1) = \{\mathbf{t}, \mathbf{f}\}, \mathit{data}(n_2) = \{\mathbf{t}, \mathbf{f}\}, \mathit{data}(n_3) = \mathbf{f}, \mathit{next}(n_1) = n_2, \mathit{next}(n_2) = \{n_1, n_3\}, \mathit{next}(n_3) = \mathit{null}$$

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

$$\mathit{pos}(A) = B, \mathit{pos}(B) = \mathit{floor}, \mathit{pos}(C) = \mathit{floor}$$

depicts the blocks world shown in Figure 1.2. The state

$$\mathit{pos}(A) = \{A, B, C, \mathit{floor}\}, \mathit{pos}(B) = \{A, B, C, \mathit{floor}\}, \mathit{pos}(C) = \{A, B, C, \mathit{floor}\}$$

signifies complete lack of information about the positions of the blocks. ■

Example 8: Consider the Hyperproof system of Example 4. The state

$$\begin{array}{lcllcllcl}
\mathit{loc}(b_1) & = & (1,1) & \mathit{size}(b_1) & = & \{\mathit{small}, \mathit{medium}\} & \mathit{shape}(b_1) & = & \mathit{cube} \\
\mathit{loc}(b_2) & = & (5,3) & \mathit{size}(b_2) & = & \mathit{small} & \mathit{shape}(b_2) & = & \mathit{tet} \\
\mathit{loc}(b_3) & = & (2,6) & \mathit{size}(b_3) & = & \mathit{large} & \mathit{shape}(b_3) & = & \{\mathit{tet}, \mathit{dodec}\} \\
\mathit{loc}(b_4) & = & \{(7,1),(7,2),\ldots,(7,8)\} & \mathit{size}(b_4) & = & \mathit{medium} & \mathit{shape}(b_4) & = & \mathit{dodec}
\end{array}$$

should be self-explanatory at this point. ■

We might think of system states as mental models of situations, representing various states of knowledge ranging from completely specific to completely general.

Definition 4: Consider a system $\mathcal{S} = (\{s_1, \ldots, s_n\}; l_1 : A_1, \ldots, l_k : A_k)$. We say that a state σ' of $\mathcal{S}$ is an **extension** of another such state σ, written $\sigma' \sqsubseteq \sigma$, iff $\sigma'(l_i, s_j) \subseteq \sigma(l_i, s_j)$ for every $i = 1, \ldots, k$ and $j = 1, \ldots, n$.[8] We call σ' a **proper extension** of σ, denoted $\sigma' \sqsubset \sigma$, iff $\sigma' \sqsubseteq \sigma$ and $\sigma \not\sqsubseteq \sigma'$. ■

Hence, σ' is a proper extension of σ 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:

$$(\{c_1, c_2\}; \mathit{hours} : \{0, \ldots, 23\}, \mathit{minutes} : \{0, \ldots, 59\}).$$

The state

$$\begin{aligned}
\mathit{hours}'(c_1) &= \{13, 14\}, \\
\mathit{minutes}'(c_1) &= \{55\}, \\
\mathit{hours}'(c_2) &= \{6, 7\}, \\
\mathit{minutes}'(c_2) &= \{9, 10\},
\end{aligned} \tag{1.1}$$

[8] The terminology sounds somewhat paradoxical, since an extension of a state is one that assigns *fewer* attribute values to each system object, thereby making our knowledge of the system more specific. This is similar to the terminology of object-oriented class hierarchies, where we say that "human" is an extension of "mammal" to mean that the former is in fact a subset of the latter.

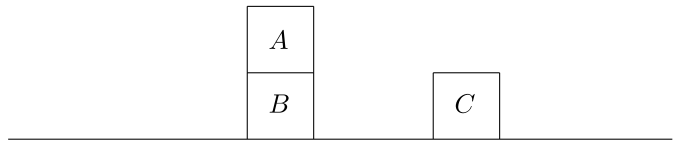


Figure 1.2: A blocks world.

is an extension of the state

$$\begin{aligned} \textit{hours}(c_1) &= \{13, 14, 15\}, \\ \textit{minutes}(c_1) &= \{55\}, \\ \textit{hours}(c_2) &= \{6, 7\}, \\ \textit{minutes}(c_2) &= \{9, 10, 11\}. \end{aligned} \tag{1.2}$$

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

$$(\mathcal{P}_\infty(A_1) \setminus \emptyset) \times \cdots \times (\mathcal{P}_\infty(A_k) \setminus \emptyset).$$

We do not have a lattice because the meet of two states might not exist. This is 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 diagrammatic ambiguity 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 σ_1 and a square shape in σ_2:

$$\begin{aligned} \sigma_1(\textit{shape}, s) &= \textit{round} \\ \sigma_2(\textit{shape}, s) &= \textit{square} \end{aligned}$$

then what is the shape of s is $\sigma_1 \sqcap \sigma_2$? If we were to define meets as

$$(\sigma_1 \sqcap \sigma_2)(l, s) = \sigma_1(l, s) \cap \sigma_2(l, s)$$

then we would have $\sigma_1 \sqcap \sigma_2(\textit{shape}, s) = \emptyset$, an impossible state of affairs. This is why some researchers have argued that diagrams are essentially an impoverished form of sentential representations (Sober 1976). Sententially, we can very well construct a formula that asserts

$$\textit{shape}(s) = \textit{round} \wedge \textit{shape}(s) = \textit{square}$$

but, diagrammatically, we cannot *draw* a square circle. This, in turn, is due to the fact that negation is not diagrammatically meaningful. If we had a negation operator $-$ on diagrams then conjunction could be defined simply as $\sigma_1 \sqcap \sigma_2 = -(\sigma_1 \sqcup \sigma_2)$. But negating a diagram could of course take us to the empty set if the starting value comprised the entire attribute space.

If we admitted a special "null diagram" indicating an inconsistent state, then we could define complementation and indeed joins and meets on diagrams, and we would obtain not just a lattice but a Boolean algebra isomorphic to sentential logic. Indeed, a mapping M from states to first-order sentences can be defined in a straightforward way, assuming we have chosen some binary predicate symbol A_l for each attribute l and appropriate constant symbols for the attribute values and system objects. We set:

$$M(\sigma) = \bigwedge_{i=1}^{n} \left[\bigwedge_{l \in L} \bigvee_{\alpha \in l(s_i)} A_l(s_i, \alpha) \right]$$

where $s_1, \ldots, s_n$ are the system objects and L the set of all labels. E.g., the conjunction of the four senteces

$$[\mathit{hours}(c_1, 13) \vee \mathit{hours}(c_1, 14)] \wedge \mathit{minutes}(c_1, 55)$$
$$[\mathit{hours}(c_2, 6) \vee \mathit{hours}(c_2, 7))] \wedge [\mathit{minutes}(c_2, 9) \vee \mathit{minutes}(c_2, 10))$$

would correspond to the state (1.1). The mapping M would then be a homomorphism:

$$\begin{array}{rcl} M(-\sigma) & = & \neg M(\sigma) \\ M(\sigma_1 \sqcup \sigma_2) & = & M(\sigma_1) \vee M(\sigma_2) \\ M(\sigma_1 \sqcap \sigma_2) & = & M(\sigma_1) \wedge M(\sigma_2) \end{array}$$

1.4 Interpreting first-order languages into system states

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

1. a relation $R^I \in \mathcal{R}$ of some arity m, called the **realization** of R:

   $$R^I \subset A_{i_1} \times \cdots \times A_{i_m}$$

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

2. a list of m pairs

   $$[(l_{i_1}, j_1), \ldots, (l_{i_m}, j_m)]$$

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

As will become apparent soon, an attribute interpretation differs from a normal interpretation in that atomic formulas over system objects are "compiled" via profiles into atomic formulas over selected attribute values of (some of) those objects. Accordingly, an atomic statement concerning system objects must be understood as an atomic statement concerning certain attribute values of those objects.

In what follows, fix a signature $\Sigma = (\mathsf{C}, \mathsf{R}, \mathsf{V})$, an attribute structure

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

and an attribute intepretation I of Σ into $\mathcal{A}$.

Suppose now that we are given an $\mathcal{A}$-system $\mathcal{S} = (\{s_1, \ldots, s_n\}; \mathcal{A})$. We define a **constant assignment** as a partial function ρ from C to $\{s_1, \ldots, s_n\}$; while a **variable assignment** is a total function χ from V to $\{s_1, \ldots, s_n\}$. We write $Dom(\rho)$ for the domain of a constant assignment ρ, i.e., the set of all and only those constant symbols for which ρ 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 ρ_1 and ρ_2 have a **conflict** iff there is some $c \in Dom(\rho_1) \cap Dom(\rho_2)$ such that $\rho_1(c) \neq \rho_2(c)$. Therefore, if $Dom(\rho_1) \supseteq Dom(\rho_2)$ then ρ_1 and ρ_2 have a conflict iff $\rho_1 \not\supseteq \rho_2$.

Formulas F over Σ 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 $t^{\rho,\chi}$ as $\rho(c)$ if t is a constant symbol c and as $\chi(v)$ if t is a variable v. Since ρ is a partial function, $t^{\rho,\chi}$ may be undefined.

By a **named state** we will mean a pair $(\sigma; \rho)$ consisting of a state σ and a constant assignment ρ. We say that a named state $(\sigma'; \rho')$ is an **extension** of a named state $(\sigma; \rho)$, written $(\sigma'; \rho') \sqsubseteq (\sigma; \rho)$, iff σ' is an extension of σ (i.e., $\sigma' \sqsubseteq \sigma$) and $\rho' \supseteq \rho$ (viewing the partial functions ρ and ρ' 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 $(\sigma'; \rho') \sqsubset (\sigma; \rho)$, iff $(\sigma'; \rho') \sqsubseteq (\sigma; \rho)$ and either $\sigma' \sqsubset \sigma$ or $\rho' \supset \rho$. 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 $(\sigma'; \rho') \overset{\infty}{\sqsubseteq} (\sigma; \rho)$ (or $(\sigma'; \rho') \overset{\infty}{\sqsubset} (\sigma; \rho)$) to indicate that $(\sigma'; \rho')$ is a finite extension (respectively, a finite proper extension) of $(\sigma; \rho)$. A named state $(\sigma; \rho)$ will be called a **world** iff σ is a world (every ascription of σ is a valuation) and ρ is total. [9] 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** β we will mean a finite set of formulas. A **context** is a pair $\gamma = (\beta; (\sigma; \rho))$ consisting of an assumption base β 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 1: *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 a named state $(\sigma; \rho)$ (of an $\mathcal{A}$-system $\mathcal{S} = (\{s_1, \ldots, s_n\}; \mathcal{A})$) along with a variable assignment χ. This is done by formally definining a mapping $I_{(\sigma;\rho)/\chi}$ from the set of all formulas to the three-element set

$$\{\textbf{true}, \textbf{false}, \textbf{unknown}\}$$

as follows.

First consider an atomic formula $R(t_1, \ldots, t_n)$, where R is a relation symbol of arity n and profile $[(l_{i_1}, j_1), \ldots, (l_{i_m}, j_m)]$. We set

$$I_{(\sigma;\rho)/\chi}(R(t_1, \ldots, t_n)) = \begin{cases} \textbf{true} & \text{if } \forall\, \alpha_1 \in l_{i_1}(t_{j_1}^{\rho,\chi}) \cdots \forall\, \alpha_m \in l_{i_m}(t_{j_m}^{\rho,\chi}) \,.\, R^I(\alpha_1, \ldots, \alpha_m); \\ \textbf{false} & \text{if } \forall\, \alpha_1 \in l_{i_1}(t_{j_1}^{\rho,\chi}) \cdots \forall\, \alpha_m \in l_{i_m}(t_{j_m}^{\rho,\chi}) \,.\, \neg R^I(\alpha_1, \ldots, \alpha_m); \\ \textbf{unknown} & \text{otherwise.} \end{cases} \quad (1.3)$$

In the first two cases above we tacitly assume—in the interest of readability—that $t_{j_x}^{\rho,\chi}$ is defined for every $x = 1, \ldots, m$. If not, then the value of $I_{(\sigma;\rho)/\chi}(R(t_1, \ldots, t_n))$ is **unknown**.

Note that the occurrences of the symbol $\forall$ on the right-hand side of (1.3) occur as part of our metalanguage and should not be confused with object-level occurrences of $\forall$ in VIVID formulas. We will continue to use

[9] This 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.

object-level symbols in different capacities without explicitly calling attention to the distinction; the context will always clarify the use.

Sentential combinations of formulas are interpreted according to the strong three-valued Kleene scheme (Kleene 1952). For instance:

$$I_{(\sigma;\,\rho)/\chi}(F_1 \wedge F_2) = \begin{cases} \textbf{true} & \text{if } I_{(\sigma;\,\rho)/\chi}(F_1) = \textbf{true} \text{ and } I_{(\sigma;\,\rho)/\chi}(F_2) = \textbf{true}; \\ \textbf{false} & \text{if } I_{(\sigma;\,\rho)/\chi}(F_1) = \textbf{false} \text{ or } I_{(\sigma;\,\rho)/\chi}(F_2) = \textbf{false}; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{1.4}$$

Finally, quantified formulas are evaluated as follows:

$$I_{(\sigma;\,\rho)/\chi}(\forall\, v \,.\, F) = \begin{cases} \textbf{true} & \text{if } I_{(\sigma;\,\rho)/\chi[v\mapsto s_i]}(F) = \textbf{true} \text{ for every } i \in \{1,\ldots,n\}; \\ \textbf{false} & \text{if } I_{(\sigma;\,\rho)/\chi[v\mapsto s_i]}(F) = \textbf{false} \text{ for some } i \in \{1,\ldots,n\}; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{1.5}$$

and

$$I_{(\sigma;\,\rho)/\chi}(\exists\, v \,.\, F) = \begin{cases} \textbf{true} & \text{if } I_{(\sigma;\,\rho)/\chi[v\mapsto s_i]}(F) = \textbf{true} \text{ for some } i \in \{1,\ldots,n\}; \\ \textbf{false} & \text{if } I_{(\sigma;\,\rho)/\chi[v\mapsto s_i]}(F) = \textbf{false} \text{ for every } i \in \{1,\ldots,n\}; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{1.6}$$

The following result is proved by a straightforward 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 2: *If $\chi_1(v) = \chi_2(v)$ for every variable v that has a free occurrence in F, then*

$$I_{(\sigma;\,\rho)/\chi_1}(F) = I_{(\sigma;\,\rho)/\chi_2}(F).$$

Lemma 3 (No unknowns in worlds): *For every world $(w; \widehat{\rho})$, variable assignment χ, and formula F,*

$$I_{(w;\,\widehat{\rho})/\chi}(F) \neq \textbf{unknown}.$$

I.e., in a world every formula is either true of false.

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

Lemma 4 (Thinning preserves truth values): *If $(\sigma'; \rho') \sqsubseteq (\sigma; \rho)$ and*

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

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

PROOF: By structural induction on F. For the basis case, suppose that F is an atomic formula $R(t_1, \ldots, t_n)$, where R has a profile $[(l_{i_1}, j_1), \ldots, (l_{i_m}, j_m)]$. From (1.3), the assumption

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

entails that the terms $t_{j_1}^{\rho,\chi}, \ldots, t_{j_m}^{\rho,\chi}$ are all defined, and further, that either

$$\forall\, \alpha_1 \in l_{i_1}(t_{j_1}^{\rho,\chi}) \cdots \forall\, \alpha_m \in l_{i_m}(t_{j_m}^{\rho,\chi}) \,.\, R^I(\alpha_1, \ldots, \alpha_m) \tag{1.7}$$

or

$$\forall\, \alpha_1 \in l_{i_1}(t^{\rho,\chi}_{j_1}) \cdots \forall\, \alpha_m \in l_{i_m}(t^{\rho,\chi}_{j_m}) \;.\; \neg R^I(\alpha_1,\ldots,\alpha_m). \tag{1.8}$$

Since $t^{\rho,\chi}_{j_1},\ldots,t^{\rho,\chi}_{j_m}$ are all defined and ρ' is a superset of ρ, the terms $t^{\rho',\chi}_{j_1},\ldots,t^{\rho',\chi}_{j_m}$ are also defined. Further, since $\sigma' \sqsubseteq \sigma$, we have

$$l'_{i_1}(t^{\rho',\chi}_{j_1}) \subseteq l_{i_1}(t^{\rho',\chi}_{j_1}),\ldots,l'_{i_m}(t^{\rho',\chi}_{j_m}) \subseteq l_{i_m}(t^{\rho',\chi}_{j_m}),$$

and since

$$t^{\rho',\chi}_{j_1} = t^{\rho,\chi}_{j_1},\ldots,t^{\rho',\chi}_{j_m} = t^{\rho,\chi}_{j_m}$$

we have

$$l'_{i_1}(t^{\rho',\chi}_{j_1}) \subseteq l_{i_1}(t^{\rho,\chi}_{j_1}),\ldots,l'_{i_m}(t^{\rho',\chi}_{j_m}) \subseteq l_{i_m}(t^{\rho,\chi}_{j_m}).$$

Hence it follows that if (1.7) is the case then

$$\forall\, \alpha_1 \in l'_{i_1}(t^{\rho',\chi}_{j_1}) \cdots \forall\, \alpha_m \in l'_{i_m}(t^{\rho',\chi}_{j_m}) \;.\; R^I(\alpha_1,\ldots,\alpha_m), \tag{1.9}$$

and therefore $I_{(\sigma';\,\rho')/\chi}(F) = I_{(\sigma;\,\rho)/\chi}(F) =$ **true**; while, if (1.8) is the case, we have

$$\forall\, \alpha_1 \in l'_{i_1}(t^{\rho',\chi}_{j_1}) \cdots \forall\, \alpha_m \in l'_{i_m}(t^{\rho',\chi}_{j_m}) \;.\; \neg R^I(\alpha_1,\ldots,\alpha_m), \tag{1.10}$$

and therefore $I_{(\sigma';\,\rho')/\chi}(F) = I_{(\sigma;\,\rho)/\chi}(F) =$ **false**. The inductive cases are straightforward. ∎

Example 9: Consider the signature $\Sigma_1 = (\mathsf{C}_{\mathit{clock}}, \mathsf{R}_{\mathit{clock}}, \mathsf{V}_{\mathit{clock}})$ where the set of constant symbols is

$$\mathsf{C}_{\mathit{clock}} = \{\mathtt{c}_1, \mathtt{c}_2, \ldots\}$$

the set of variables is $\mathsf{V}_{\mathit{clock}} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$, and the set of relation symbols is

$$\mathsf{R}_{\mathit{clock}} = \{\mathtt{PM}, \mathtt{AM}, \mathtt{Ahead}, \mathtt{Behind}\},$$

with `PM`, `AM` unary and `Ahead`, `Behind` binary.

Consider now the attribute structure

$$\mathit{Clock} = (\mathit{hours} : \{0,\ldots,23\}, \mathit{minutes} : \{0,\ldots,59\}; \{R_1, R_2, R_3, R_4\}),$$

where $R_1 \subseteq \mathit{hours}$, $R_2 \subseteq \mathit{hours}$,

$$R_3 \subseteq \mathit{hours} \times \mathit{minutes} \times \mathit{hours} \times \mathit{minutes},$$

$$R_4 \subseteq \mathit{hours} \times \mathit{minutes} \times \mathit{hours} \times \mathit{minutes},$$

defined as follows: $R_1(h) \Leftrightarrow h > 11$, $R_2(h) \Leftrightarrow h \leq 11$,

$$R_3(h_1, m_1, h_2, m_2) \Leftrightarrow h_1 > h_2 \vee (h_1 = h_2 \wedge m_1 > m_2),$$

and

$$R_4(h_1, m_1, h_2, m_2) \Leftrightarrow h_1 \leq h_2 \vee (h_1 = h_2 \wedge m_1 \leq m_2).$$

We define an interpretation I of Σ_1 into this attribute structure by specifying a unique relation (in the structure) and a unique profile for each symbol in $\mathsf{R}_{\mathit{clock}}$. In particular, we set $\mathtt{PM}^I = R_1, \mathtt{AM}^I = R_2, \mathtt{Ahead}^I = R_3, \mathtt{Behind}^I = R_4$ and:

$$\begin{aligned}
\mathit{Prof}(\mathtt{PM}) &= [(\mathit{hours}, 1)];\\
\mathit{Prof}(\mathtt{AM}) &= [(\mathit{hours}, 1)];\\
\mathit{Prof}(\mathtt{Ahead}) &= [(\mathit{hours}, 1), (\mathit{minutes}, 1), (\mathit{hours}, 2), (\mathit{minutes}, 2)];\\
\mathit{Prof}(\mathtt{Behind}) &= [(\mathit{hours}, 1), (\mathit{minutes}, 1), (\mathit{hours}, 2), (\mathit{minutes}, 2)].
\end{aligned}$$

∎

Example 10: Consider the system $(\{c_1, c_2\}; \mathit{Clock})$, where *Clock* is the attribute structure of Example 9. Let σ be the following state of this system:

$$\begin{array}{rcl} \mathit{hours}(c_1) & = & \{9, 13\}, \\ \mathit{minutes}(c_1) & = & 12, \\ \mathit{hours}(c_2) & = & 8, \\ \mathit{minutes}(c_2) & = & 27, \end{array}$$

and let ρ be the partial constant assignment that maps $\mathtt{c}_1$ to c_1 and $\mathtt{c}_2$ to c_2. We claim that the sentence $\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)$ is true in $(\sigma; \rho)$ for any variable assignment χ. Indeed, consider an arbitrary χ. Recalling the profile of `Ahead`, definition (1.3) tells us that in order to have

$$I_{(\sigma;\rho)/\chi}(\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)) = \textbf{true}$$

we must have $R(a_1, a_2, a_3, a_4)$ for all

$$\begin{array}{rcl} a_1 & \in & \mathit{hours}(\mathtt{c}_1^{\rho,\chi} = c_1) = \{9, 13\}, \\ a_2 & \in & \mathit{minutes}(\mathtt{c}_1^{\rho,\chi} = c_1) = \{12\}, \\ a_3 & \in & \mathit{hours}(\mathtt{c}_2^{\rho,\chi} = c_2) = \{8\}, \\ a_4 & \in & \mathit{minutes}(\mathtt{c}_2^{\rho,\chi} = c_2) = \{27\}, \end{array}$$

i.e., we must have $R_3(9, 12, 8, 27)$ and $R_3(13, 12, 8, 27)$. Both of these hold according to the definition of R_3, since $9 > 8$ and $13 > 8$.

As another example, the sentence $\mathtt{PM}(\mathtt{c}_1) \wedge \neg\mathtt{PM}(\mathtt{c}_1)$ evaluates to **unknown** in $(\sigma; \rho)$, despite being patently inconsistent, because, intuitively, in $(\sigma; \rho)$we do not know whether $\mathtt{c}_1$ is prior to midnight or after it (one possibility is 9, which is a.m., and the other is 13, which is p.m.). Therefore, $\mathtt{PM}(\mathtt{c}_1)$ evaluates to **unknown**, hence $\neg\mathtt{PM}(\mathtt{c}_1)$ also evaluates to **unknown**, and therefore their conjunction evaluates to **unknown** as well. It is instructive to see why, precisely, $\mathtt{PM}(\mathtt{c}_1)$ evaluates to **unknown**. Recall that the interpretation of `PM` is the unary relation R_1, which holds of a given hour h iff $h > 11$; and that the profile of `PM` is $[(\mathit{hours}, 1)]$. Accordingly, for $\mathtt{PM}(\mathtt{c}_1)$ to be true in $(\sigma; \rho)$ (and arbitrary χ), we must have $R_1(a_1)$ for all

$$a_1 \in \mathit{hours}(\mathtt{c}_1^{\rho,\chi} = c_1) = \{9, 13\},$$

i.e., we must have $9 > 11$ and $13 > 11$, which is clearly false. Likewise, for $\mathtt{PM}(\mathtt{c}_1)$ to come out **false** in $(\sigma; \rho)$ and χ, we must have $\neg R_1(9)$ and $\neg R_1(13)$, which is also false. Accordingly,

$$I_{(\sigma;\rho)/\chi}(\mathtt{PM}(\mathtt{c}_1)) = \textbf{unknown}$$

by (1.3). ■

The following is a direct consequence of the finite size of the ascription values, the finite number of system objects, and Lemma 2.

Lemma 5: *$I_{(\sigma;\rho)/\chi}$ is computable for any named state $(\sigma; \rho)$ and variable assignment χ.*

Definition 5: A world $(w; \widehat{\rho})$ **satisfies a formula** F w.r.t. a variable assignment χ iff

$$I_{(w;\widehat{\rho})/\chi}(F) = \textbf{true}.$$

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

Lemma 6 (Weakening): *If* $(\beta;(\sigma;\rho)) \models F$ *then* $(\beta \cup \beta';(\sigma;\rho)) \models F$*; and if* $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$ *then* $(\beta \cup \beta';(\sigma;\rho)) \models (\sigma';\rho')$.

Lemma 7: *If* $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$ *and* $(\beta;(\sigma';\rho')) \models F$ *then* $(\beta;(\sigma;\rho)) \models F$.

PROOF: Pick any world $(w;\widehat{\rho})$ and variable assignment χ and suppose that

$$(w;\widehat{\rho}) \models_\chi (\beta;(\sigma;\rho)). \tag{1.11}$$

Then, by the assumption $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$, we conclude

$$(w;\widehat{\rho}) \models (\sigma';\rho'). \tag{1.12}$$

From (1.11) and (1.12) we infer

$$(w;\widehat{\rho}) \models_\chi (\beta;(\sigma';\rho')). \tag{1.13}$$

Finally, (1.13) and the assumption $(\beta;(\sigma';\rho')) \models F$ imply $(w;\widehat{\rho}) \models_\chi F$. ■

Lemma 8: $(\beta;(\sigma;\rho)) \models (\sigma;\rho)$.

Lemma 9: $(\beta \cup \{\textbf{false}\};(\sigma;\rho)) \models (\sigma';\rho')$.

PROOF: Pick any world $(w;\widehat{\rho})$ and variable assignment χ, and assume

$$(w;\widehat{\rho}) \models_\chi (\beta \cup \{\textbf{false}\};(\sigma;\rho)),$$

so that $(w;\widehat{\rho}) \models_\chi$ **false**. But, by definition,

$$I_{(w;\widehat{\rho})/\chi}(\textbf{false}) = \textbf{false},$$

and the contradiction entitles us to infer $(w;\widehat{\rho}) \models (\sigma';\rho')$. ■

Lemma 10: *If* $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$ *and* $(\sigma';\rho') \sqsubseteq (\sigma'';\rho'')$ *then* $(\beta;(\sigma;\rho)) \models (\sigma'';\rho'')$.

PROOF: Pick any world $(w;\widehat{\rho})$ and variable assignment χ and suppose that

$$(w;\widehat{\rho}) \models_\chi (\beta;(\sigma;\rho)), \tag{1.14}$$

so that

$$(w;\widehat{\rho}) \sqsubseteq (\sigma;\rho) \tag{1.15}$$

and

$$I_{(w;\widehat{\rho})/\chi}(F) = \textbf{true} \tag{1.16}$$

for all $F \in \beta$. From the assumption $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$ and (1.14) we obtain $(w;\widehat{\rho}) \models (\sigma';\rho')$, which is to say $(w;\widehat{\rho}) \sqsubseteq (\sigma';\rho')$. Finally, $(w;\widehat{\rho}) \sqsubseteq (\sigma';\rho')$, the assumption $(\sigma';\rho') \sqsubseteq (\sigma'';\rho'')$ and Lemma 1 yield $(w;\widehat{\rho}) \sqsubseteq (\sigma'';\rho'')$, i.e., $(w;\widehat{\rho}) \models (\sigma'';\rho'')$. ■

Corollary 11 (Widening is sound): *If* $(\sigma;\rho) \sqsubseteq (\sigma';\rho')$ *then* $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$.

PROOF: By Lemma 8, $(\beta;(\sigma;\rho)) \models (\sigma;\rho)$, hence, by Lemma 10, $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$. ∎

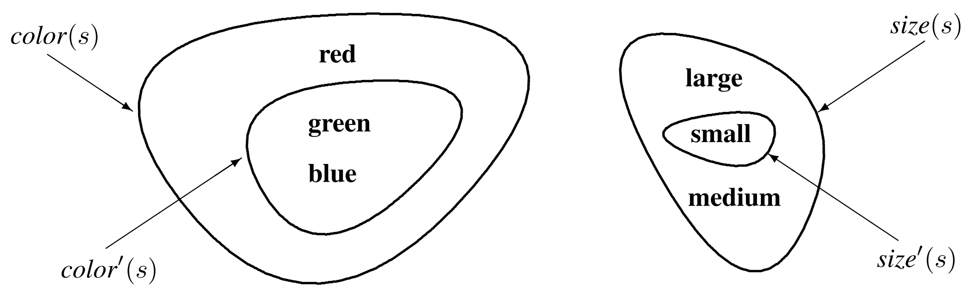
Next we formalize the important notion of alternative extensions.

Definition 6: Let σ_1, σ_2 be proper extensions of a state σ. We say that σ_2 is an **alternative extension** of σ with respect to σ_1, written $Alt(\sigma, \sigma_1, \sigma_2)$, 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 $\sigma_2(l', s') = \sigma(l', s')$. ∎

It follows immediately that if such an atribute 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 σ 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 σ_1 extends σ by limiting the color values of s to **green** and **blue** and its size to **small**:



What counts as an alternative extension of σ w.r.t. σ_1? Considering that σ_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:

Color of s	*Size* of s
{**red**}	{**large**, **medium**}
{**red**}	{**small**}
{**green**, **blue**}	{**large**, **medium**}
{**red**}	{**large**, **medium**}

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 memberhip represents disjunctive information, we can collapse the first two and last two possibilities, obtaining:

Color of s	*Size* of s
{**red**}	{**small**, **large**, **medium**}
{**red**, **green**, **blue**}	{**large**, **medium**}

These are the only two alternative extensions of σ w.r.t. σ_1. In general, given an arbitrary extension $\sigma_1 \sqsubset \sigma$, we can effectively construct all alternative extensions of σ w.r.t. σ_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 σ to σ_1. We can generate the alternative states by taking the complement of the ascription value of each such pair in σ_1 [10] (clause 2 of Definition 6) while reverting the other $m-1$ pairs to their σ values (clause 3 of Definition 6).

We stress that in determining the alternative extensions of σ w.r.t. σ_1 we only consider those objects and attributes that are changed by σ_1. We ignore those ascription assignments that remain the same in going from σ to σ_1. As another example, there are two states that are alternative extensions of (1.2) w.r.t. (1.1). 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 (1.2), as neither of them was modified by (1.1).

Lemma 12: *If $w, \sigma' \sqsubset \sigma$ and $w \not\sqsubseteq \sigma'$ then there is some $\sigma'' \sqsubset \sigma$ such that $Alt(\sigma, \sigma', \sigma'')$ and $w \sqsubseteq \sigma''$. In words: if a world w and a state σ' both extend σ and w does not extend σ', then there is an alternative extension σ'' of σ w.r.t. σ' such that w extends σ''.*

PROOF: Since both σ' and w are extensions of σ, we have

$$\sigma'(l,s) \subseteq \sigma(l,s) \tag{1.17}$$

and

$$w(l,s) \subseteq \sigma(l,s) \tag{1.18}$$

for every attribute l and system object s. Further, since $w \not\sqsubseteq \sigma'$, 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 α such that

$$\alpha \in w(l_i, s_j) \tag{1.19}$$

and

$$\alpha \notin \sigma'(l_i, s_j). \tag{1.20}$$

Moreover, since w is a world we have $w(l_i, s_j) = \{\alpha\}$. From (1.19) and (1.18) we infer

$$\alpha \in \sigma(l_i, s_j). \tag{1.21}$$

From (1.21), (1.20), and (1.17) we obtain

$$\sigma'(l_i, s_j) \subset \sigma(l_i, s_j). \tag{1.22}$$

Now define $\sigma'' \sqsubset \sigma$ as follows:

$$\sigma''(l_i, s_j) = \sigma(l_i, s_j) \setminus \sigma'(l_i, s_j) \tag{1.23}$$

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

$$\sigma''(l,s) = \sigma(l,s). \tag{1.24}$$

It follows by construction (specifically, from (1.22), (1.23), and (1.24)) that $Alt(\sigma, \sigma', \sigma'')$. Further, $w \sqsubseteq \sigma''$. 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 (1.23), (1.21), and (1.20) 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 (1.24) and (1.18). ■

[10] The complement with respect to the corresponding ascription value in σ.

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 9 below) subsumes the one given above.

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 σ 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. σ 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

$$\left[\bigcup_{j=1}^{m'} \sigma_{i_j}(l_{i_j}, s_{i_j})\right] \subset \sigma(l_{i_1}, s_{i_1}).$$

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

$$\sigma_{i_1}(l_{i_1}, s_{i_1}) \cup \cdots \cup \sigma_{i_{m'}}(l_{i_{m'}}, s_{i_{m'}}) = \sigma(l_{i_1}, s_{i_1}).$$ ■

Note that every list of length one that spans σ_1 w.r.t. σ (for $\sigma_1 \sqsubset \sigma$) does so properly. That is why the definition below is a proper generalization of Definition 6.

Definition 9: Let $\sigma_1, \ldots, \sigma_m, \sigma' \sqsubset \sigma$, $m \geq 1$. We say that σ' is an **alternative extension of** σ **w.r.t.** $\sigma_1, \ldots, \sigma_m$, written $\mathit{Alt}(\sigma, \{\sigma_1, \ldots, \sigma_m\}, \sigma')$, 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. σ such that for every attribute l and object s we have

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

We write $\mathbf{A}(\{\sigma_1, \ldots, \sigma_m\}, \sigma)$ for the set of all alternative extensions of σ w.r.t. $\sigma_1, \ldots, \sigma_m$. ■

Therefore, to compute all alternative extensions of σ 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. σ. We will present algorithms for both tasks shortly, but we first turn to an example that will help to clarify these definitions.

Example 11: Suppose we have two objects s_1 and s_2, and two attributes, color and size, with color having three possible values: red, green and blue (abbreviated R, G, and B); and where size has three possible values: small, medium, and large (ab. S, M, and L). Suppose further that the starting state σ is as follows:

σ
$\mathit{Color}(s_1) = \{R, B\}$
$\mathit{Size}(s_1) = \{S, M, L\}$
$\mathit{Color}(s_2) = \{R, B, G\}$
$\mathit{Size}(s_2) = \{M, L\}$

Now consider the following three proper extensions of σ:

σ_1	σ_2	σ_3
$\mathit{Color}(s_1) = \{B\}$ **A**	$\mathit{Color}(s_1) = \{R, B\}$	$\mathit{Color}(s_1) = \{R\}$ **F**
$\mathit{Size}(s_1) = \{S, M\}$ **B**	$\mathit{Size}(s_1) = \{L\}$ **D**	$\mathit{Size}(s_1) = \{S, M, L\}$
$\mathit{Color}(s_2) = \{B, G\}$ **C**	$\mathit{Color}(s_2) = \{R, B, G\}$	$\mathit{Color}(s_2) = \{R, B, G\}$
$\mathit{Size}(s_2) = \{M, L\}$	$\mathit{Size}(s_2) = \{L\}$ **E**	$\mathit{Size}(s_2) = \{M, L\}$

We have used the labels **A—F** to mark those a.o. pairs $(l; s)$ for which state σ_i properly extends σ, i.e., such that $\sigma_i(l, s) \subset \sigma(l, s)$. The following lists of a.o. pairs span σ_1, σ_2, and σ_3 w.r.t. σ:

$L_1 = [(\mathit{Color}; s_1)\ \ (\mathit{Size}; s_1)\ \ (\mathit{Color}; s_1)]$ (corresponding to **A-D-F**)

$L_2 = [(\mathit{Color}; s_1)\ \ (\mathit{Size}; s_2)\ \ (\mathit{Color}; s_1)]$ **(A-E-F)**

$L_3 = [(\mathit{Size}; s_1)\ \ (\mathit{Size}; s_1)\ \ (\mathit{Color}; s_1)]$ **(B-D-F)**

$L_4 = [(\mathit{Size}; s_1)\ \ (\mathit{Size}; s_2)\ \ (\mathit{Color}; s_1)]$ **(B-E-F)**

$L_5 = [(\mathit{Color}; s_2)\ \ (\mathit{Size}; s_1)\ \ (\mathit{Color}; s_1)]$ **(C-D-F)**

$L_6 = [(\mathit{Color}; s_2)\ \ (\mathit{Size}; s_2)\ \ (\mathit{Color}; s_1)]$ **(C-E-F)**

These are the only lists that span σ_1, σ_2, and σ_3 w.r.t. σ. From these, only L_4, L_5, and L_6 do so properly. L_1 does not span σ_1, σ_2, and σ_3 properly (w.r.t. σ) because $[1\ 3]$ is a sublist of $[1\ 2\ 3]$ such that $[L_1(1)\ \ L_1(3)]$, namely $[(\mathit{Color}; s_1)\ \ (\mathit{Color}; s_1)]$, is homogeneous and yet

$$\sigma_1(\mathit{Color}, s_1) \cup \sigma_3(\mathit{Color}, s_1) = \{R, B\} \not\subset \sigma(\mathit{Color}, s_1) = \{R, B\}.$$

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

$$[L_3(1)\ \ L_3(2)] = [(\mathit{Size}; s_1)\ \ (\mathit{Size}; s_1)]$$

is homogeneous but

$$\sigma_1(\mathit{Size}, s_1) \cup \sigma_2(\mathit{Size}, s_1) = \{S, M, L\} \not\subset \sigma(\mathit{Size}, s_1) = \{S, M, L\}.$$

Accordingly, we have a total of three alternative extensions of σ w.r.t. σ_1, σ_2, and σ_3, corresponding to L_4, L_5, and L_6:

σ_4 **(B-E-F)**	σ_5 **(C-D-F)**	σ_6 **(C-E-F)**
$\mathit{Color}(s_1) = \{B, G\}$	$\mathit{Color}(s_1) = \{B, G\}$	$\mathit{Color}(s_1) = \{B, G\}$
$\mathit{Size}(s_1) = \{L\}$	$\mathit{Size}(s_1) = \{S, M\}$	$\mathit{Size}(s_1) = \{S, M, L\}$
$\mathit{Color}(s_2) = \{R, B, G\}$	$\mathit{Color}(s_2) = \{R\}$	$\mathit{Color}(s_2) = \{R\}$
$\mathit{Size}(s_2) = \{M\}$	$\mathit{Size}(s_2) = \{M, L\}$	$\mathit{Size}(s_2) = \{M\}$

so that

$$\mathbf{A}(\{\sigma_1, \sigma_2, \sigma_3\}, \sigma) = \{\sigma_4, \sigma_5, \sigma_6\}.$$

Thus each alternative state is uniquely determined by the corresponding list that properly spans σ_1, σ_2, and σ_3 w.r.t. σ. Specifically, the ascription values of each alternative state are obtained by "flipping" (complementing) the corresponding ascription values of σ 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 σ are retained, since in those cases we have $\mathit{Pos}((l;s),L)=\emptyset$ and hence

$$\sigma(l,s) \setminus \bigcup_{i \in \mathit{Pos}((l;s),L)} \sigma_i(l,s) = \sigma(l,s).$$

Intuitively, this ensures that every alternative state has a maximal disagreement with each extension of σ. ■

The following algorithm computes the set of lists that properly span states $\sigma_1,\ldots,\sigma_m$ w.r.t. σ:

1. Let Φ 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 Ψ be the set obtained from Φ^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,
   $$\Psi = \{[(l_1;s_1)\cdots(l_m;s_m)] \in \Phi^m \mid \sigma_i(l_i,s_i) \subset \sigma(l_i,s_i) \text{ for } i=1,\ldots,m\}.$$

   Thus Ψ is the set of all and only those lists that span $\sigma_1,\ldots,\sigma_m$ w.r.t. σ.

3. From Ψ, filter out all those lists that do not properly span $\sigma_1,\ldots,\sigma_m$ w.r.t. σ, and return the result. To determine whether a list $[(l_1;s_1)\cdots(l_m;s_m)]$ in Ψ properly spans $\sigma_1,\ldots,\sigma_m$ w.r.t. σ, 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 $i=1,\ldots,m$:
     $f \leftarrow f[(l_i;s_i) \mapsto f(l_i,s_i) \cup \{i\}]$;
     $P \leftarrow P \cup \{(l_i;s_i)\}$.
   - For each pair $(l;s) \in P$: if

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

     return *false*, else continue.
   - Return *true*.

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

1. Let Ψ be the set of all and only those lists of a.o. pairs that properly span $\sigma_1,\ldots,\sigma_m$ w.r.t. σ.

2. Let $\Sigma \leftarrow \emptyset$.

3. For each list $L \in \Psi$:

   - Let σ' be the unique state such that for any l and s,

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

- $\Sigma \leftarrow \Sigma \cup \{\sigma'\}$.

4. Return Σ.

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

Lemma 13: *$Alt(\sigma, \sigma', \sigma'')$ iff $Alt(\sigma, \{\sigma'\}, \sigma'')$.*

The following result generalizes Lemma 12.

Lemma 14: *If $\sigma_1, \ldots, \sigma_m, w \sqsubset \sigma$ and $w \not\sqsubseteq \sigma_i$ for every $i = 1, \ldots, m$, then there is some $\sigma' \sqsubset \sigma$ such that $Alt(\sigma, \{\sigma_1, \ldots, \sigma_m\}, \sigma')$ and $w \sqsubseteq \sigma'$.*

PROOF: By assumption, we have

$$\forall\, l, s \,.\, w(l, s) \subseteq \sigma(l, s); \tag{1.25}$$

$$\forall\, i \in \{1, \ldots, m\} \,.\, \forall\, l, s \,.\, \sigma_i(l, s) \subseteq \sigma(l, s). \tag{1.26}$$

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 α_i such that

$$w(l_i, s_i) = \{\alpha_i\} \tag{1.27}$$

and

$$\alpha_i \notin \sigma_i(l_i, s_i). \tag{1.28}$$

From (1.27) and (1.25) we infer

$$\forall\, i \in \{1, \ldots, m\} \,.\, \alpha_i \in \sigma(l_i, s_i). \tag{1.29}$$

Hence, from (1.28) and (1.26) we conclude

$$\forall\, i \in \{1, \ldots, m\} \,.\, \sigma_i(l_i, s_i) \subset \sigma(l_i, s_i). \tag{1.30}$$

Therefore, the list

$$L = [(l_1; s_1) \cdots (l_m; s_m)]$$

spans $\sigma_1, \ldots, \sigma_m$ w.r.t. σ. 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

$$\alpha_{i_1} = \cdots = \alpha_{i_{m'}}. \tag{1.31}$$

Now suppose, by way of contradiction, that

$$\bigcup_{j=1}^{m'} \sigma_{i_j}(l_{i_j}, s_{i_j}) = \sigma(l_{i_1}, s_{i_1}). \tag{1.32}$$

From (1.27) and (1.25) we conclude

$$w(l_{i_1}, s_{i_1}) = \{\alpha_{i_1}\} \subseteq \sigma(l_{i_1}, s_{i_1}),$$

so that

$$\alpha_{i_1} \in \sigma(l_{i_1}, s_{i_1}). \tag{1.33}$$

Hence, by (1.32),

$$\alpha_{i_1} \in \bigcup_{j=1}^{m'} \sigma_{i_j}(l_{i_j}, s_{i_j}),$$

which means that there is some $j \in \{1, \ldots, m'\}$ such that

$$\alpha_{i_1} \in \sigma_{i_j}(l_{i_j}, s_{i_j}). \tag{1.34}$$

From (1.28) we get $\alpha_{i_j} \notin \sigma_{i_j}(l_{i_j}, s_{i_j})$. But, by (1.31), $\alpha_{i_1} = \alpha_{i_j}$, hence $\alpha_{i_1} \notin \sigma_{i_j}(l_{i_j}, s_{i_j})$, contradicting (1.34). Therefore, L spans $\sigma_1, \ldots, \sigma_m$ w.r.t. σ properly.

Now define $\sigma' \sqsubset \sigma$ as follows: for any l and s,

$$\sigma'(l, s) = \sigma(l, s) \setminus \bigcup_{i \in \mathit{Pos}((l;s), L)} \sigma_i(l, s). \tag{1.35}$$

By construction, $\mathit{Alt}(\sigma, \{\sigma_1, \ldots, \sigma_m\}, \sigma')$. Further, we have $w \sqsubseteq \sigma'$. 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 $\mathit{Pos}((l; s), L) = \emptyset$ and hence, from (1.35), $\sigma'(l, s) = \sigma(l, s)$, so $w(l, s) \subseteq \sigma'(l, s)$ follows from (1.25). Suppose, by contrast, that $(l; s)$ occurs in L, so that

$$\mathit{Pos}((l; s), L) = \{i_1, \ldots, i_{m'}\}$$

for some m' such that $1 \le m' \le m$. From (1.28),

$$\forall\, j \in \{1, \ldots, m'\} \,.\, \alpha_{i_j} \notin \sigma_{i_j}(l_{i_j}, s_{i_j}). \tag{1.36}$$

But

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

and since

$$(l_{i_1}; s_{i_1}) = \cdots = (l_{i_{m'}}; s_{i_{m'}}) = (l; s), \tag{1.37}$$

we get $\alpha_{i_1} = \cdots = \alpha_{i_{m'}}$. Accordingly, (1.36) yields

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

which, by virtue of (1.37), becomes

$$\forall\, j \in \{1, \ldots, m'\} \,.\, \alpha_{i_1} \notin \sigma_{i_j}(l, s). \tag{1.38}$$

It follows from (1.38) that

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

or, equivalently,

$$\alpha_{i_1} \notin \bigcup_{i \in \mathit{Pos}((l;s),L)} \sigma_i(l,s). \tag{1.39}$$

However,

$$\alpha_{i_1} \in \sigma(l_{i_1}, s_{i_1}) = \sigma(l,s), \tag{1.40}$$

and hence we infer from (1.39), (1.40), and (1.35) that

$$\alpha_{i_1} \in \sigma'(l,s). \tag{1.41}$$

But, from (1.37), $w(l_{i_1}, s_{i_1}) = w(l,s)$, which is to say $w(l,s) = \{\alpha_{i_1}\}$. We have thus shown that, in this case too, $w(l,s) \subseteq \sigma'(l,s)$. ∎

We now extend the notion of alternative extensions to named states.

Definition 10: Let $(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m),(\sigma';\rho') \stackrel{\infty}{\sqsubset} (\sigma;\rho)$, $m \geq 1$. 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

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

iff $\mathit{Dom}(\rho') = \mathit{Dom}(\rho_1) \cup \cdots \cup \mathit{Dom}(\rho_m)$ and there is a subset $S \subseteq \{1,\ldots,m\}$ such that:

1. ρ' conflicts with ρ_i iff $i \in S$; and
2. if $S \neq \{1,\ldots,m\}$ then $\mathit{Alt}(\sigma,\{\sigma_i \mid i \in \{1,\ldots,m\} \setminus S\},\sigma')$. ∎

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 $\mathit{Alt}((\sigma;\rho),(\sigma_1;\rho_1),(\sigma';\rho'))$ instead of $\mathit{Alt}((\sigma;\rho),\{(\sigma_1;\rho_1)\},(\sigma';\rho'))$.

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

1. Let $\rho'_1,\ldots,\rho'_k$, $k \geq 1$, be all and only the constant assignments on $\mathit{Dom}(\rho_1) \cup \cdots \cup \mathit{Dom}(\rho_m)$ that are supersets of ρ. Note that there are $k = n^d$ such assignments, where n is the number of system objects and
$$d = |[\mathit{Dom}(\rho_1) \cup \cdots \cup \mathit{Dom}(\rho_m)] \setminus \mathit{Dom}(\rho)|.$$
2. Let $R = \emptyset$.
3. For each ρ'_i, $i = 1,\ldots,k$, do the following:
   - Let $\Sigma_i \subseteq \{\sigma_1,\ldots,\sigma_m\}$ consist of all and only those states $\sigma_j, j \in \{1,\ldots,m\}$ such that ρ'_i does *not* have a conflict with ρ_j, meaning that $\rho'_i \supseteq \rho_j$.
   - Let $\Phi_i = \mathbf{A}(\Sigma_i,\sigma)$.
   - 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 $\mathbf{A}(\Sigma_i,\sigma)$ for the various i. Memorizing intermediate results and building $\mathbf{A}(\Sigma_i,\sigma)$ incrementally could improve its efficiency.

Lemma 15: *If* $(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m) \stackrel{\infty}{\sqsubset} (\sigma;\rho)$, $m \geq 1$, $(w;\widehat{\rho}) \sqsubseteq (\sigma;\rho)$, *and*

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

then there is $(\sigma';\rho') \stackrel{\infty}{\sqsubset} (\sigma;\rho)$ *such that* $\mathit{Alt}((\sigma;\rho),\{(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m)\},(\sigma';\rho'))$ *and* $(w;\widehat{\rho}) \sqsubseteq (\sigma';\rho')$.

PROOF: The following holds by assumption:

$$\forall\, i \in \{1,\ldots,m\}\, .\, w \not\sqsubseteq \sigma_i \vee\, \widehat{\rho} \not\supseteq \rho_i. \tag{1.42}$$

Define

$$S = \{i \in \{1,\ldots,m\} \mid \widehat{\rho} \not\supseteq \rho_i\} \tag{1.43}$$

and let

$$\rho' = \widehat{\rho} \restriction [\mathit{Dom}(\rho_1) \cup \cdots \cup \mathit{Dom}(\rho_m)], \tag{1.44}$$

so that

$$\widehat{\rho} \supseteq \rho' \supseteq \rho. \tag{1.45}$$

It follows by construction that

$$\mathit{Dom}(\rho') = \mathit{Dom}(\rho_1) \cup \cdots \cup \mathit{Dom}(\rho_m)$$

and

$$\forall\, i \in \{1,\ldots,m\}\, .\, \rho' \not\supseteq \rho_i \Leftrightarrow i \in S,$$

which is to say that ρ' has a conflict with ρ_i iff $i \in S$. At this point there are two cases: $S = \{1,\ldots,m\}$ or $S \subset \{1,\ldots,m\}$. In the first case, we must have

$$\rho' \supset \rho, \tag{1.46}$$

for if $\rho' = \rho$ then, from (1.44), $\rho_1 = \cdots = \rho_m = \rho$ and hence $\widehat{\rho} \supseteq \rho_i$ for all $i = 1,\ldots,m$, since $\rho \subseteq \widehat{\rho}$ by assumption. But, from (1.43), $\forall i \in \{1,\ldots,m\}\, .\, \widehat{\rho} \supseteq \rho_i$ would entail $S = \emptyset$, contradicting the supposition $S = \{1,\ldots,m\}$ (recall that $m \geq 1$). Define $\sigma' = w$. Then $(\sigma';\rho') \sqsubseteq (\sigma;\rho)$ by (1.45), and indeed $(\sigma';\rho') \overset{\infty}{\sqsubset} (\sigma;\rho)$ by (1.46) and (1.44). In addition, $(w;\widehat{\rho}) \sqsubseteq (\sigma';\rho')$ follows from $w \sqsubseteq w$ and (1.45).

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

$$\{1,\ldots,m\} \setminus S \neq \emptyset. \tag{1.47}$$

From the definition of S and (1.42) we infer

$$\forall\, i \in \{1,\ldots,m\} \setminus S\, .\, w \not\sqsubseteq \sigma_i. \tag{1.48}$$

From (1.48) we can infer that

$$\forall\, i \in \{1,\ldots,m\} \setminus S\, .\, \sigma_i \sqsubset \sigma, \tag{1.49}$$

for otherwise there would be some $j \in \{1,\ldots,m\} \setminus S$ such that $\sigma_j \not\sqsubset \sigma$ and $\sigma_j \sqsubseteq \sigma$, and hence $\sigma_j = \sigma$. But $w \sqsubseteq \sigma = \sigma_j$ contradicts the assumption $w \not\sqsubseteq \sigma_j$. Further, we can infer $w \sqsubset \sigma$, for, in light of (1.47), $w = \sigma$ would contradict (1.49), given that worlds do not have any proper extensions. Therefore, by Lemma 14, there exists a state

$$\sigma' \sqsubset \sigma \tag{1.50}$$

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

$$w \sqsubseteq \sigma'. \tag{1.51}$$

From (1.50) and (1.45) we conclude $(\sigma';\rho') \overset{\infty}{\sqsubset} (\sigma;\rho)$. Further, by construction,

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

while $(w;\widehat{\rho}) \sqsubseteq (\sigma';\rho')$ follows from (1.51) and (1.45). This concludes the case analysis. ■

Corollary 16: *If* $(\sigma'; \rho') \overset{\infty}{\sqsubset} (\sigma; \rho)$, $(w; \widehat{\rho}) \sqsubseteq (\sigma; \rho)$, *and* $(w; \widehat{\rho}) \not\sqsubseteq (\sigma'; \rho')$ *then there is some*

$$(\sigma''; \rho'') \overset{\infty}{\sqsubset} (\sigma; \rho)$$

such that $Alt((\sigma; \rho), (\sigma'; \rho'), (\sigma''; \rho''))$ *and* $(w; \widehat{\rho}) \sqsubseteq (\sigma''; \rho'')$.

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

Definition 11: Suppose that $(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m) \sqsubset (\sigma; \rho)$ and let β be any assumption base. We say that $(\sigma; \rho)$ **entails** $(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)$ **w.r.t.** β, written $(\sigma; \rho) \Vdash\!\vdash_\beta \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}$, iff for every $(\sigma'; \rho')$ such that

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

there is some $F \in \beta$ such that, for all χ,

$$I_{(\sigma'; \rho')/\chi}(F) = \textbf{false}. \qquad \blacksquare$$

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

This definition captures the intuition that any world which extends the state $(\sigma; \rho)$ and satisfies the formulas in β 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 β. (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 17 below.

Determining whether or not $(\sigma; \rho) \Vdash\!\vdash_\beta \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}$ 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

$$\forall\, i \in \{1, \ldots, k\} \;.\; \chi_i \restriction FV(F) = \psi_i.$$

3. If $I_{(\sigma; \rho)/\chi_i} = \textbf{false}$ for every $i = 1, \ldots, k$ then return *true*, else return *false*.

The algorithm for determining $(\sigma; \rho) \Vdash\!\vdash_\beta \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}$ 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 $\exists\, F \in \beta \;.\; g(F, (\sigma'; \rho'))$ then continue, else return *false*.

2. Return *true*.

The algorithm clearly hinges on g, whose correctness in this context depends on Lemma 2.

Lemma 17: *If* $(\sigma; \rho) \Vdash\!\vdash_\beta \{(\sigma_1; \rho_1), \ldots, (\sigma_m; \rho_m)\}$ *then for all worlds* $(w; \widehat{\rho})$ *and variable assignments* χ, *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

$$(\sigma;\rho) \Vdash\!\vdash_\beta \{(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m)\} \tag{1.52}$$

pick any world $(w;\widehat{\rho})$ and variable assignment χ and suppose that $(w;\widehat{\rho}) \models_\chi (\beta;(\sigma;\rho))$ so that

$$(w;\widehat{\rho}) \sqsubseteq (\sigma;\rho) \tag{1.53}$$

and

$$\forall\, F \in \beta \,.\, I_{(w;\widehat{\rho})/\chi}(F) = \textbf{true}. \tag{1.54}$$

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.,

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

By Lemma 15, there is some

$$(\sigma';\rho') \overset{\infty}{\sqsubset} (\sigma;\rho)$$

such that

$$Alt((\sigma;\rho),\{(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m)\},(\sigma';\rho')) \tag{1.55}$$

and

$$(w;\widehat{\rho}) \sqsubseteq (\sigma';\rho'). \tag{1.56}$$

But then, by Definition 11 and (1.55) it follows that there is some

$$G \in \beta \tag{1.57}$$

such that

$$I_{(\sigma';\rho')/\chi}(G) = \textbf{false}, \tag{1.58}$$

and hence, from the Thinning Lemma (Lemma 4) in tandem with (1.56) and (1.58), we obtain

$$I_{(w;\widehat{\rho})/\chi}(G) = \textbf{false},$$

which contradicts (1.54) in view of (1.57). ■

Corollary 18: *If* $(\sigma;\rho) \Vdash\!\vdash_\beta (\sigma';\rho')$ *then* $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$.

1.5 A family of diagrammatic natural deduction languages

We now introduce 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 $\Sigma = (\mathsf{C},\mathsf{R},\mathsf{V})$, an attribute structure $\mathcal{A} = (\{l_1 : A_1,\ldots,l_k : A_k\};\mathcal{R})$, and an interpretation I of R into $\mathcal{A}$. We assume in what follows that Σ, $\mathcal{A}$, and I have been fixed. The terms and formulas of the language are defined as described in Section 1.4. 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.

Lemma 19: *If* $b \in \{\textbf{true},\textbf{false}\}$,

$$I_{(\sigma;\rho)/\chi[v\,\mapsto\,s]}(F) = b$$

and v' *does not occur in* F *then*

$$I_{(\sigma;\rho)/\chi[v'\,\mapsto\,s]}(F\,[v'/v]) = b.$$

1.5.1 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 Δ to range over sentential and diagrammatic deductions, respectively. The symbol $\mathfrak{D}$ will range over the union of the two. The abstract syntax (Reynolds 1998) of both proof types is defined by the grammars below:

$$
\begin{array}{lcl}
D & ::= & \textit{RuleApp} \\
 & | & \textbf{assume } F\ D \\
 & | & F\ \textbf{ by }\ D \\
 & | & \mathfrak{D}; D \\
 & | & \textbf{pick-any } x\ \ D \\
 & | & \textbf{pick-witness } w\ \ \textbf{for}\ \exists\, x \,.\, F\ \ D \\
 & | & \textbf{specialize } \forall\, x_1 \cdots x_n \,.\, F \textbf{ with } t_1, \ldots, t_n \\
 & | & \textbf{ex-generalize } \exists\, x \,.\, F \textbf{ from } t \\
 & | & \textbf{cases from } F_1, \ldots, F_k\textbf{:}\ (\sigma_1; \rho_1) \ \rightarrow\ D_1 \mid \cdots \mid (\sigma_n; \rho_n) \ \rightarrow\ D_n \\
 & | & \textbf{observe }\ F
\end{array}
$$

$$
\begin{array}{lcl}
\Delta & ::= & \mathfrak{D}; \Delta \\
 & | & \textbf{claim } (\sigma; \rho) \\
 & | & (\sigma; \rho) \textbf{ by thinning with } F_1, \ldots, F_n \\
 & | & (\sigma; \rho) \textbf{ by widening} \\
 & | & (\sigma; \rho) \textbf{ by absurdity} \\
 & | & \textbf{cases from } F_1, \ldots, F_k\textbf{:}\ (\sigma_1; \rho_1) \ \rightarrow\ \Delta_1 \mid \cdots \mid (\sigma_n; \rho_n) \ \rightarrow\ \Delta_n \\
 & | & \textbf{cases } F_1 \ \vee\ F_2\textbf{:}\ F_1 \ \rightarrow\ \Delta_1 \mid F_2 \ \rightarrow\ \Delta_2 \\
 & | & \textbf{pick-witness } w\ \ \textbf{for}\ \exists\, x \,.\, F\ \ \Delta
\end{array}
$$

$$
\mathfrak{D} \ ::= \ D \mid \Delta
$$

where the syntax of inference *rule applications* is as follows:

$$
\begin{array}{lcl}
\textit{RuleApp} & ::= & \textbf{claim }\ F \\
 & | & \textbf{true-intro} \\
 & | & \textbf{modus-ponens } F \ \Rightarrow\ G, F \\
 & | & \textbf{modus-tollens } F \ \Rightarrow\ G, \neg G \\
 & | & \textbf{double-negation } \neg\neg F \\
 & | & \textbf{absurd } F, \neg F \\
 & | & \textbf{left-and } F \ \wedge\ G \\
 & | & \textbf{right-and } F \ \wedge\ G \\
 & | & \textbf{both } F, G \\
 & | & \textbf{left-either } F, G \\
 & | & \textbf{right-either } F, G \\
 & | & \textbf{cases } F_1 \ \vee\ F_2, F_1 \ \Rightarrow\ G, F_2 \ \Rightarrow\ G \\
 & | & \textbf{left-iff } F \ \Leftrightarrow\ G \\
 & | & \textbf{right-iff } F \ \Leftrightarrow\ G \\
 & | & \textbf{equiv } F \ \Rightarrow\ G, G \ \Rightarrow\ F
\end{array}
$$

The composition operator ";" associates to the right by default, so $\mathfrak{D}_1; \mathfrak{D}_2; \mathfrak{D}_3$ stands for

$$\mathfrak{D}_1; (\mathfrak{D}_2; \mathfrak{D}_3)$$

rather than $(\mathfrak{D}_1; \mathfrak{D}_2); \mathfrak{D}_3$. Parentheses or **begin-end** pairs can be used to change the default grouping.

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

$$\begin{array}{rcl}
(\mathfrak{D}_1; \mathfrak{D}_2)\,[t/x] & = & \mathfrak{D}_1\,[t/x]; \mathfrak{D}_2\,[t/x] \\
((\sigma;\rho)\ \textbf{by thinning with}\ F_1,\ldots,F_n)\,[t/x] & = & (\sigma;\rho)\ \textbf{by thinning with}\ F_1\,[t/x],\ldots,F_n\,[t/x] \\
\begin{array}{c}(\textbf{cases from}\ F_1,\ldots,F_k\textbf{:} \\ (\sigma_1;\rho_1)\ \rightarrow\ \Delta_1 \mid \cdots \mid (\sigma_n;\rho_n)\ \rightarrow\ \Delta_n)\,[t/x]\end{array} & = & \begin{array}{c}\textbf{cases from}\ F_1\,[t/x],\ldots,F_k\,[t/x]\textbf{:} \\ (\sigma_1;\rho_1)\ \rightarrow\ \Delta_1\,[t/x] \mid \cdots \mid (\sigma_n;\rho_n)\ \rightarrow\ \Delta_n\,[t/x]\end{array} \\
\begin{array}{c}(\textbf{cases}\ F_1\ \vee\ F_2\textbf{:} \\ F_1\ \rightarrow\ \Delta_1 \mid F_2\ \rightarrow\ \Delta_2)[t/x]\end{array} & = & \begin{array}{c}\textbf{cases}\ F_1\,[t/x]\ \vee\ F_2\,[t/x]\textbf{:} \\ F_1\,[t/x]\ \rightarrow\ \Delta_1\,[t/x] \mid F_2\,[t/x]\ \rightarrow\ \Delta_2\,[t/x]\end{array} \\
(\textbf{pick-witness}\ x\ \ \textbf{for}\ \exists\, y\,.\,F\ \ \Delta)[t/x] & = & \textbf{pick-witness}\ x\ \ \textbf{for}\ (\exists\, y\,.\,F)\,[t/x]\ \ \Delta \\
\begin{array}{c}(\textbf{pick-witness}\ w\ \ \textbf{for}\ \exists\, y\,.\,F\ \ \Delta)[t/x] \\ (\text{when } x \neq w)\end{array} & = & \textbf{pick-witness}\ w\ \ \textbf{for}\ (\exists\, y\,.\,F)\,[t/x]\ \ \Delta\,[t/x] \\
(\textbf{pick-any}\ x\ \ D)\,[t/x] & = & \textbf{pick-any}\ x\ \ D \\
\begin{array}{c}(\textbf{pick-any}\ y\ \ D)\,[t/x] \\ (\text{when } x \neq y)\end{array} & = & \textbf{pick-any}\ y\ \ D\,[t/x]
\end{array}$$

We omit the defining equations for the sentential **pick-witness**, which is handled like the diagrammatic **pick-witness**; and for the remaining **cases from**, 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 α-renamed away from the given term t.

1.5.2 Evaluation semantics

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

$$\gamma \vdash D \leadsto F$$

and

$$\gamma \vdash \Delta \leadsto (\sigma; \rho)$$

which are read as:

"In the context γ, deduction D (Δ) derives F (respectively, $(\sigma;\rho)$)."

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

$$\frac{}{(\beta \cup \{F \wedge G\}; (\sigma;\rho)) \vdash \textbf{left-and}\ F \wedge G \leadsto F}$$

$$\frac{}{(\beta \cup \{F_1,\ldots,F_n\};(\sigma;\rho)) \vdash (\sigma';\rho')\ \textbf{by thinning with}\ F_1,\ldots,F_n \leadsto (\sigma';\rho')} \quad [\textit{Thinning}]$$

provided $(\sigma;\rho) \Vvdash_{\{F_1,\ldots,F_n\}} (\sigma';\rho')$

$$\frac{}{(\beta;(\sigma;\rho)) \vdash (\sigma';\rho')\ \textbf{by widening} \leadsto (\sigma';\rho')} \quad [\textit{Widening}]$$

provided $(\sigma;\rho) \sqsubseteq (\sigma';\rho')$

$$\frac{}{(\beta \cup \{\textbf{false}\};(\sigma;\rho)) \vdash (\sigma';\rho')\ \textbf{by absurdity} \leadsto (\sigma';\rho')} \quad [\textit{Absurdity}]$$

$$\frac{}{(\beta;(\sigma;\rho)) \vdash \textbf{claim}\ (\sigma;\rho) \leadsto (\sigma;\rho)} \quad [\textit{Diagram-Reitaration}]$$

$$\frac{\begin{array}{c}(\beta \cup \{F_1,\ldots,F_k\};(\sigma_1;\rho_1)) \vdash \Delta_1 \leadsto (\sigma';\rho') \\ \vdots \\ (\beta \cup \{F_1,\ldots,F_k\};(\sigma_n;\rho_n)) \vdash \Delta_n \leadsto (\sigma';\rho')\end{array}}{(\beta \cup \{F_1,\ldots,F_k\};(\sigma;\rho)) \vdash \textbf{cases from}\ F_1,\ldots,F_k\textbf{:}\ (\sigma_1;\rho_1) \ \to\ \Delta_1 \mid \cdots \mid (\sigma_n;\rho_n) \ \to\ \Delta_n \leadsto (\sigma';\rho')} \quad [C_1]$$

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

$$\frac{(\beta \cup \{F_1 \vee F_2, F_1\};(\sigma;\rho)) \vdash \Delta_1 \leadsto (\sigma';\rho') \qquad (\beta \cup \{F_1 \vee F_2, F_2\};(\sigma;\rho)) \vdash \Delta_2 \leadsto (\sigma';\rho')}{(\beta \cup \{F_1 \vee F_2;(\sigma;\rho)) \vdash \textbf{cases}\ F_1 \vee F_2\ \ F_1 \rightarrow \Delta_1 \mid F_2 \rightarrow \Delta_2 \leadsto (\sigma';\rho')} \quad [C_2]$$

$$\frac{(\beta;(\sigma;\rho)) \vdash D \leadsto F \qquad (\beta \cup \{F\};(\sigma;\rho)) \vdash \Delta \leadsto (\sigma';\rho')}{(\beta;(\sigma;\rho)) \vdash D;\Delta \leadsto (\sigma';\rho')} \quad [D;\Delta]$$

$$\frac{(\beta;(\sigma;\rho)) \vdash \Delta \leadsto (\sigma';\rho') \qquad (\beta;(\sigma';\rho')) \vdash D \leadsto F}{(\beta;(\sigma;\rho)) \vdash \Delta;D \leadsto F} \quad [\Delta;D]$$

$$\frac{(\beta;(\sigma;\rho)) \vdash \Delta_1 \leadsto (\sigma_1;\rho_1) \qquad (\beta;(\sigma_1;\rho_1)) \vdash \Delta_2 \leadsto (\sigma_2;\rho_2)}{(\beta;(\sigma;\rho)) \vdash \Delta_1;\Delta_2 \leadsto (\sigma_2;\rho_2)} \quad [\Delta;\Delta]$$

$$\frac{(\beta;(\sigma;\rho)) \vdash D_1 \leadsto F_1 \qquad (\beta \cup \{F_1\};(\sigma;\rho)) \vdash D_2 \leadsto F_2}{(\beta;(\sigma;\rho)) \vdash D_1;D_2 \leadsto F_2} \quad [D;D]$$

$$\frac{(\beta \cup \{\exists\, x \,.\, F, F\,[z/x]\};(\sigma;\rho)) \vdash \Delta\,[z/w] \leadsto (\sigma';\rho')}{(\beta \cup \{\exists\, x \,.\, F\};(\sigma;\rho)) \vdash \textbf{pick-witness}\ w\ \ \textbf{for}\ \exists\, x \,.\, F\ \ \Delta \leadsto (\sigma';\rho')} \quad [\textit{EI}/\Delta]$$

provided z is fresh

Figure 1.3: Formal semantics of diagrammatic deductions

$$\frac{(\beta \cup \{F\}; (\sigma; \rho)) \vdash D \leadsto G}{(\beta; (\sigma; \rho)) \vdash \textbf{assume } F \;\; D \leadsto F \Rightarrow G}$$

The only new sentential forms are **observe**, **cases from**, and $\Delta; D$. We will discuss the last two later; the semantics of **observe** are as follows:

$$\frac{}{(\beta; (\sigma; \rho)) \vdash \textbf{observe } F \leadsto F} \quad [\textit{Observe}]$$

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

The side condition is computable because of Lemma 5 and because, by Lemma 2, we need only 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 $F_1 \vee F_2$ 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:

$$\textbf{cases } F_1 \vee F_2, F_1 \Rightarrow G, F_2 \Rightarrow G.$$

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

$$\frac{}{(\beta \cup \{F_1 \vee F_2, F_1 \Rightarrow G, F_2 \Rightarrow G\}; (\sigma; \rho)) \vdash \textbf{cases } F_1 \vee F_2, F_1 \Rightarrow G, F_2 \Rightarrow G \leadsto G}$$

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

$$\textbf{cases } F_1 \;\vee\; F_2\textbf{:}\;\; F_1 \;\rightarrow\; \Delta_1 \mid F_2 \;\rightarrow\; \Delta_2,$$

which is classified as a diagrammatic deduction ("a Δ") since the end result is a diagram. The semantics of this form are given by rule $[C_2]$, shown in Figure 1.3.

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, 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

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

This is classified as a sentential deduction, since the end result is a formula F. Its semantics are shown in Figure 1.4. 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

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

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

$$\frac{\begin{array}{c}(\beta \cup \{F_1,\ldots,F_k\}; (\sigma_1;\rho_1)) \vdash D_1 \leadsto F \\ \vdots \\ (\beta \cup \{F_1,\ldots,F_k\}; (\sigma_n;\rho_n)) \vdash D_n \leadsto F\end{array}}{\begin{array}{rl}(\beta \cup \{F_1,\ldots,F_k\}; (\sigma;\rho)) \vdash \textbf{cases from } F_1,\ldots,F_k\textbf{:} & (\sigma_1;\rho_1) \to D_1 \mid \\ & \cdots \\ & (\sigma_n;\rho_n) \to D_n \leadsto F\end{array}} \quad [C_3]$$

$$\text{provided } (\sigma;\rho) \Vdash\!\vdash_{\{F_1,\ldots,F_k\}} \{(\sigma_1;\rho_1),\ldots,(\sigma_n;\rho_n)\}$$

Figure 1.4: Semantics of diagrammatic-to-sentential case reasoning.

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:

$$\textbf{cases from } F_1,\ldots,F_k\textbf{:}\ (\sigma_1;\rho_1) \to \Delta_1 \mid \cdots \mid (\sigma_n;\rho_n) \to \Delta_n.$$

The corresponding semantics are given by rule $[C_1]$, shown in Figure 1.3.

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 1.3. They are isomorphic to the regular composition semantics of $\mathcal{NDL}$, 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 conlusion of Δ. Its semantics are prescribed by rule $[D; \Delta]$. Observe that the conclusion of D becomes available to Δ (e.g., the conclusion of D could be a disjunction and Δ might be a diagrammatic case analysis of that disjunction).

3. $\Delta; D$, where a diagrammatic deduction Δ is composed with a sentential deduction. This form is classified as a sentential deduction since the end result is a formula (the conlusion 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 Δ. E.g., D might be an **observe** deduction that points out something that can be seen in the diagram derived by Δ.

4. $\Delta_1; \Delta_2$, where a diagrammatic deduction Δ_1 is composed with another diagrammatic deduction Δ_2. This form is of course classified as a diagrammatic deduction, since the end result is a diagram (the conlusion of Δ_2). Its semantics are given by rule $[\Delta; \Delta]$. The same principle of conclusion threading applies here: Δ_2 is evaluated in the system state resulting from the evaluation of Δ_1; the assumption base is threaded through unchanged.

Theorem 20 (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.[11] We will omit most sentential forms, as those have been proved sound elsewhere (Arkoudas 2000).

[11] To be perfectly precise, we are proving the statement: "For all positive integers n and for all γ, D, Δ, 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 $\gamma \vdash D \leadsto F$ then $\gamma \models (\sigma;\rho)$. It is readily seen that this statement implies Theorem 20.

The basis cases correspond to the axioms of our semantics. In what follows we will treat the diagrammatic axioms [*Observe*], [*Absurdity*], [*Diagram-Reitaration*], [*Widening*], and [*Thinning*].

- [*Observe*]: In this case D is of the form **observe** F and we need to show that

$$(\beta; (\sigma; \rho)) \models F$$

whenever $(\beta; (\sigma; \rho)) \vdash D \rightsquigarrow F$. To that end, consider an arbitrary world $(w; \widehat{\rho})$ and variable assignment χ and suppose that $(w; \widehat{\rho}) \models_{\chi} (\beta; (\sigma; \rho))$, so that

$$(w; \widehat{\rho}) \sqsubseteq (\sigma; \rho). \tag{1.59}$$

By the side condition of [*Observe*], it must be that

$$I_{(\sigma;\,\rho)/\chi}(F) = \textbf{true},$$

and hence, from (1.59) and Lemma 4,

$$I_{(w;\,\widehat{\rho})/\chi}(F) = \textbf{true},$$

which is to say $(w; \widehat{\rho}) \models_{\chi} F$. We have thus shown that $(w; \widehat{\rho}) \models_{\chi} (\beta; (\sigma; \rho))$ implies $(w; \widehat{\rho}) \models_{\chi} F$ for any $(w; \widehat{\rho})$ and χ, which establishes $(\beta; (\sigma; \rho)) \models F$.

- [*Thinning*]: Here Δ is of the form

$$(\sigma'; \rho') \textbf{ by thinning with } F_1, \ldots, F_n$$

and we need to show that if

$$(\beta \cup \{F_1, \ldots, F_n\}; (\sigma; \rho)) \vdash \Delta \rightsquigarrow (\sigma'; \rho') \tag{1.60}$$

then

$$(\beta \cup \{F_1, \ldots, F_n\}; (\sigma; \rho)) \models (\sigma'; \rho'). \tag{1.61}$$

From (1.60) and the side condition of [*Thinning*] we obtain

$$(\sigma; \rho) \Vdash\!\vdash_{\{F_1, \ldots, F_n\}} (\sigma'; \rho'),$$

and hence, by Corollary 18, $(\{F_1, \ldots, F_n\}; (\sigma; \rho)) \models (\sigma'; \rho')$. Now (1.61) follows from weakening (Lemma 6).

- [*Widening*]: Here Δ is of the form

$$(\sigma'; \rho') \textbf{ by widening}$$

and we must show that $(\beta; (\sigma; \rho)) \models (\sigma'; \rho')$ whenever $(\beta; (\sigma; \rho)) \vdash \Delta \rightsquigarrow (\sigma'; \rho')$. From the side condition of [*Widening*] we infer $(\sigma; \rho) \sqsubseteq (\sigma'; \rho')$, and now the desired $(\beta; (\sigma; \rho)) \models (\sigma'; \rho')$ follows from Corollary 11.

- [*Diagram-Reitaration*]: Here the result follows directly from Lemma 8.

- [*Absurdity*]: Here Δ is of the form

$$(\sigma'; \rho') \textbf{ by absurdity}$$

and we need to show

$$(\beta \cup \{\textbf{false}\}; (\sigma; \rho)) \models (\sigma'; \rho')$$

whenever $(\beta \cup \{\textbf{false}\}; (\sigma; \rho)) \vdash \Delta \rightsquigarrow (\sigma'; \rho')$. This follows from Lemma 9.

- $[C_1]$: Here Δ is of the form

$$\textbf{cases from } F_1, \ldots, F_k\textbf{:}\ (\sigma_1; \rho_1) \ \rightarrow \ \Delta_1 \mid \cdots \mid (\sigma_n; \rho_n) \ \rightarrow \ \Delta_n.$$

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

$$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \vdash \Delta \rightsquigarrow (\sigma'; \rho'). \tag{1.62}$$

We need to show

$$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \models (\sigma'; \rho'). \tag{1.63}$$

From (1.62) and $[C_1]$ we infer

$$\forall\, i \in \{1, \ldots, n\} \,.\, (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_i; \rho_i)) \vdash \Delta_i \rightsquigarrow (\sigma'; \rho') \tag{1.64}$$

and

$$(\sigma; \rho) \Vdash\!\!\vdash_{\{F_1, \ldots, F_k\}} \{(\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)\}. \tag{1.65}$$

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

$$(w; \widehat{\rho}) \models_\chi (\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \tag{1.66}$$

so that

$$(w; \widehat{\rho}) \models_\chi (\{F_1, \ldots, F_k\}; (\sigma; \rho)). \tag{1.67}$$

From (1.65), Lemma 17, and (1.67) we conclude that $(w; \widehat{\rho}) \models (\sigma_j; \rho_j)$ for some $j \in \{1, \ldots, n\}$. By the inductive hypothesis, (1.64) yields

$$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_j; \rho_j)) \models (\sigma'; \rho'), \tag{1.68}$$

and since

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

it follows from (1.68) that $(w; \widehat{\rho}) \models (\sigma'; \rho')$.

- $[C_2]$: Here Δ is of the form

$$\textbf{cases } F_1 \vee F_2 \ \ F_1 \rightarrow \Delta_1 \mid F_2 \rightarrow \Delta_2$$

and, assuming

$$(\beta \cup \{F_1 \vee F_2\}; (\sigma; \rho)) \vdash \Delta \rightsquigarrow (\sigma'; \rho'), \tag{1.69}$$

we need to show

$$(\beta \cup \{F_1 \vee F_2\}; (\sigma; \rho)) \models (\sigma'; \rho'). \tag{1.70}$$

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

$$(w; \widehat{\rho}) \models_\chi (\beta \cup \{F_1 \vee F_2\}; (\sigma; \rho)) \tag{1.71}$$

so that

$$I_{(w; \widehat{\rho})/\chi}(F_1) = \textbf{true} \tag{1.72}$$

or

$$I_{(w; \widehat{\rho})/\chi}(F_2) = \textbf{true} \tag{1.73}$$

(note that this inference would not be sanctioned in a weak three-valued Kleene logic). Now from (1.69) and $[C_2]$ we get

$$(\beta \cup \{F_1 \vee F_2, F_1\}; (\sigma; \rho)) \vdash \Delta_1 \rightsquigarrow (\sigma'; \rho') \tag{1.74}$$

and

$$(\beta \cup \{F_1 \vee F_2, F_2\}; (\sigma; \rho)) \vdash \Delta_2 \rightsquigarrow (\sigma'; \rho'). \tag{1.75}$$

Inductively, (1.74) and (1.75) respectively yield

$$(\beta \cup \{F_1 \vee F_2, F_1\}; (\sigma; \rho)) \models (\sigma'; \rho') \tag{1.76}$$

and

$$(\beta \cup \{F_1 \vee F_2, F_2\}; (\sigma; \rho)) \models (\sigma'; \rho'). \tag{1.77}$$

Now if (1.72) holds then, from (1.71), we have

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

and hence $(w; \widehat{\rho}) \models (\sigma'; \rho')$ follows from (1.76); while if (1.73) holds then

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

and hence $(w; \widehat{\rho}) \models (\sigma'; \rho')$ follows from (1.77). Therefore, $(w; \widehat{\rho}) \models (\sigma'; \rho')$ holds in either case.

- $[C_3]$: Here Δ is of the form

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

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

  $$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \vdash \Delta \rightsquigarrow F, \tag{1.78}$$

  so that

  $$\forall\, i \in \{1, \ldots, n\} \,.\, (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_i; \rho_i)) \vdash D_i \rightsquigarrow F \tag{1.79}$$

  and

  $$(\sigma; \rho) \Vdash\!\vdash_{\{F_1, \ldots, F_k\}} \{(\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)\}. \tag{1.80}$$

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

  $$(w; \widehat{\rho}) \models_\chi (\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)). \tag{1.81}$$

  It follows that

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

  and hence by Lemma 17 and (1.80) we conclude that $(w; \widehat{\rho}) \sqsubseteq (\sigma_j; \rho_j)$ for some $j \in \{1, \ldots, n\}$. Inductively, from (1.79), we infer

  $$(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_j; \rho_j)) \models F. \tag{1.82}$$

  But from $(w; \widehat{\rho}) \sqsubseteq (\sigma_j; \rho_j)$ and (1.81) we get

  $$(w; \widehat{\rho}) \models_\chi (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_i; \rho_i)),$$

  and therefore (1.82) yields $(w; \widehat{\rho}) \models_\chi F$.

- [*EI*/Δ]: In that case the deduction is of the form

$$\textbf{pick-witness}\ w\ \ \textbf{for}\ \exists\, x\,.\,F\ \ \Delta$$

and assuming that

$$(\beta \cup \{\exists\, x\,.\,F\}; (\sigma;\rho)) \vdash \textbf{pick-witness}\ w\ \ \textbf{for}\ \exists\, x\,.\,F\ \ \Delta \rightsquigarrow (\sigma';\rho'), \tag{1.83}$$

we need to show

$$(\beta \cup \{\exists\, x\,.\,F\}; (\sigma;\rho)) \models (\sigma';\rho'). \tag{1.84}$$

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

$$(w;\widehat{\rho}) \models_{\chi} (\beta \cup \{\exists\, x\,.\,F\}; (\sigma;\rho)). \tag{1.85}$$

From (1.83) and the [EI/Δ] rule we infer that, for some fresh variable z,

$$(\beta \cup \{\exists\, x\,.\,F, F\,[z/x]\}; (\sigma;\rho)) \vdash \Delta\,[z/w] \rightsquigarrow (\sigma';\rho'). \tag{1.86}$$

From (1.86) and the inductive hypothesis we obtain

$$(\beta \cup \{\exists\, x\,.\,F, F\,[z/x]\}; (\sigma;\rho)) \models (\sigma';\rho'). \tag{1.87}$$

From (1.85) and (1.6) we conclude that there is some system object s such that

$$I_{(w;\widehat{\rho})/\chi[x \,\mapsto\, s]}(F) = \textbf{true}.$$

Therefore, from Lemma 19,

$$I_{(w;\widehat{\rho})/\chi[z \,\mapsto\, s]}(F\,[z/x]) = \textbf{true},$$

and since z does not occur in $\beta \cup \{\exists\, x\,.\,F\}$, we also have (by (1.85) and Lemma 2):

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

Hence,

$$(w;\widehat{\rho}) \models_{\chi[z \,\mapsto\, s]} \beta \cup \{\exists\, x\,.\,F, F\,[z/x]\}, \tag{1.88}$$

and since $(w;\widehat{\rho}) \sqsubseteq (\sigma;\rho)$ (from (1.85)), we conclude that

$$(w;\widehat{\rho}) \models_{\chi[z \,\mapsto\, s]} (\beta \cup \{\exists\, x\,.\,F, F\,[z/x]\}; (\sigma;\rho)). \tag{1.89}$$

Finally, from (1.89) and (1.87) we obtain $(w;\widehat{\rho}) \models (\sigma';\rho')$.

- [$D;\Delta$]: Here the deduction is of the form $D;\Delta$, and assuming that

$$(\beta; (\sigma;\rho)) \vdash D;\Delta \rightsquigarrow (\sigma';\rho'), \tag{1.90}$$

we need to show

$$(\beta; (\sigma;\rho)) \models (\sigma';\rho'). \tag{1.91}$$

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

$$(w;\widehat{\rho}) \models_{\chi} (\beta; (\sigma;\rho)). \tag{1.92}$$

From (1.90) and the $[D; \Delta]$ rule we infer that, for some F,

$$(\beta; (\sigma; \rho)) \vdash D \rightsquigarrow F, \tag{1.93}$$

$$(\beta \cup \{F\}; (\sigma; \rho)) \vdash \Delta \rightsquigarrow (\sigma'; \rho'). \tag{1.94}$$

From (1.93) and the inductive hypothesis we obtain $(\beta; (\sigma; \rho)) \models F$, which, in tandem with (1.92), yields

$$(w; \widehat{\rho}) \models_\chi F.$$

Therefore,

$$(w; \widehat{\rho}) \models_\chi (\beta \cup \{F\}; (\sigma; \rho)). \tag{1.95}$$

Now (1.94) and the inductive hypothesis give

$$(\beta \cup \{F\}; (\sigma; \rho)) \models (\sigma'; \rho'), \tag{1.96}$$

and finally (1.95) and (1.96) produce the desired $(w; \widehat{\rho}) \models_\chi (\sigma'; \rho')$.

- $[\Delta; D]$: Here the proof is of the form $\Delta; D$ and assuming that

$$(\beta; (\sigma; \rho)) \vdash \Delta; D \rightsquigarrow F, \tag{1.97}$$

we need to show $(\beta; (\sigma; \rho)) \models F$. Accordingly, consider any world $(w; \widehat{\rho})$ and variable assignment χ such that

$$(w; \widehat{\rho}) \models_\chi (\beta; (\sigma; \rho)). \tag{1.98}$$

From (1.97) and the $[\Delta; D]$ rule we conclude that, for some $(\sigma'; \rho')$,

$$(\beta; (\sigma; \rho)) \vdash \Delta \rightsquigarrow (\sigma'; \rho') \tag{1.99}$$

and

$$(\beta; (\sigma'; \rho')) \vdash D \rightsquigarrow F. \tag{1.100}$$

From (1.99) and the inductive hypothesis we get $(\beta; (\sigma; \rho)) \models (\sigma'; \rho')$, so (1.98) yields

$$(w; \widehat{\rho}) \models (\sigma'; \rho')$$

and hence

$$(w; \widehat{\rho}) \models_\chi (\beta; (\sigma'; \rho')). \tag{1.101}$$

But from (1.100) and the inductive hypothesis we get $(\beta; (\sigma'; \rho')) \models F$, which, along with (1.101), entails $(w; \widehat{\rho}) \models_\chi F$.

- $[\Delta; \Delta]$: Here the deduction is of the form $\Delta_1; \Delta_2$. Assuming

$$(\beta; (\sigma; \rho)) \vdash \Delta_1; \Delta_2 \rightsquigarrow (\sigma_2; \rho_2), \tag{1.102}$$

we must show $(\beta; (\sigma; \rho)) \models (\sigma_2; \rho_2)$. Pick any $(w; \widehat{\rho})$ and χ such that

$$(w; \widehat{\rho}) \models_\chi (\beta; (\sigma; \rho)). \tag{1.103}$$

From (1.102) and rule $[\Delta; \Delta]$ we infer that, for some $(\sigma_1; \rho_1)$,

$$(\beta; (\sigma; \rho)) \vdash \Delta_1 \rightsquigarrow (\sigma_1; \rho_1); \tag{1.104}$$

$$(\beta; (\sigma_1; \rho_1)) \vdash \Delta_2 \rightsquigarrow (\sigma_2; \rho_2). \tag{1.105}$$

From (1.104), (1.105), and the inductive hypotheses we get

$$(\beta; (\sigma; \rho)) \models (\sigma_1; \rho_1); \tag{1.106}$$

$$(\beta; (\sigma_1; \rho_1)) \models (\sigma_2; \rho_2). \tag{1.107}$$

From (1.103) and (1.106) we infer $(w; \widehat{\rho}) \models (\sigma_1; \rho_1)$, so that

$$(w; \widehat{\rho}) \models_\chi (\beta; (\sigma_1; \rho_1)),$$

which in tandem with (1.107) yields the desired $(w; \widehat{\rho}) \models (\sigma_2; \rho_2)$.

This completes the case analysis and the induction. ■

Example 12: Consider the VIVID language obtained by fixing the clock signature, attribute structure and interpretation of Example 9. Now consider a system of two clocks c_1 and c_2, to which we will give the names $\mathtt{c}_1$ and $\mathtt{c}_2$ (recall that $\mathtt{c}_1$ and $\mathtt{c}_2$ are constant symbols of the signature, so this is a constant assignment ρ, which need only be partial). Now let σ be the state depicted by the following picture:

$$\underset{\mathtt{c}_1}{\boxed{\{4,5,6\}\text{:}28}} \qquad\qquad \underset{\mathtt{c}_2}{\boxed{5\text{:}45}}$$

Intuitively, this state signifies that we know the precise time displayed by c_2 (5:45 am). We are also sure of the minute value of c_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 $\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)$, indicating that the time displayed by c_1 is ahead of that displayed by c_2.

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

$$\underset{\mathtt{c}_1}{\boxed{6\text{:}28}} \qquad\qquad \underset{\mathtt{c}_2}{\boxed{5\text{:}45}}$$

That is, we should be able to conclude the exact time of c_1, since, given that c_1 is ahead of c_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:

$$(\sigma'; \rho) \ \textbf{by thinning with}\ \mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2).$$

This deduction, when evaluated in the context $(\{\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)\}; (\sigma; \rho))$, will result in the state (diagram) $(\sigma'; \rho)$. More formally, we have the following judgment:

$$(\{\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)\}; (\sigma; \rho)) \vdash (\sigma'; \rho) \ \textbf{by thinning with}\ \mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2) \rightsquigarrow (\sigma'; \rho)$$

by virtue of

$$(\sigma; \rho) \Vdash\!\vdash_{\{\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)\}} (\sigma'; \rho). \tag{1.108}$$

Note that ρ does not change in the resulting state.

To establish (1.108) rigorously, we must show that for all named states $(\sigma''; \rho'')$ such that

$$\mathit{Alt}((\sigma; \rho), (\sigma'; \rho), (\sigma''; \rho''))$$

we have

$$I_{(\sigma'';\rho'')/\chi}(\texttt{Ahead}(\texttt{c}_1,\texttt{c}_2)) = \textbf{false}$$

for all variable assignments χ, according to Definition 11. Given that the assignment ρ does not change, it follows from Definition 10 that we must have $\rho'' = \rho$ and hence $\mathit{Alt}(\sigma, \sigma', \sigma'')$. Now there is only one alternative extension σ'' of σ w.r.t. σ', obtained from σ by complementing the *hours* value of c_1 in σ' with respect to the corresponding value in σ:

$$\sigma'' : \mathit{hours}(c_1) = \{4, 5\}.$$

It is straightforward to verify that

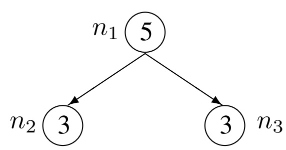
$$I_{(\sigma'';\rho)/\chi}(\texttt{Ahead}(\texttt{c}_1,\texttt{c}_2)) = \textbf{false}$$

for all χ. ■

1.6 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 1.7.

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 *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 *children* $: N \to N^*$ (arbitrary lists can be chosen if the ordering is immaterial for displaying the graph). Consider, for instance, the graph:



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 *data* and *children* as mentioned above, where

$$\mathit{data}(n_1) = 5, \mathit{data}(n_2) = 3, \mathit{data}(n_3) = 3$$

and

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

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 $\mathcal{S}_N = (N; \mathcal{A}_N)$, where $\mathcal{A}_N$ is an automorphic attribute structure of the form

$$\mathcal{A}_N = (\mathit{id} : N, \mathit{children} : N^*, \mathit{data} : V; \mathcal{R}).$$

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

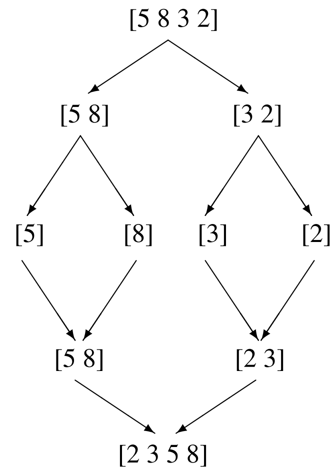


Figure 1.5: The call graph resulting from the application of MergeSort to the list [5 8 3 2].

1.7 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 1.7.2 we formalize it rigorously as an instance of VIVID.

1.7.1 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 1.5 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 1.6, 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 1.6 along with what we know about Mergesort, 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 1.6, 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 1.7. 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 1.7. That part consists of six steps, labeled (1) through (6). The new information extracted by each step appears in red 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_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

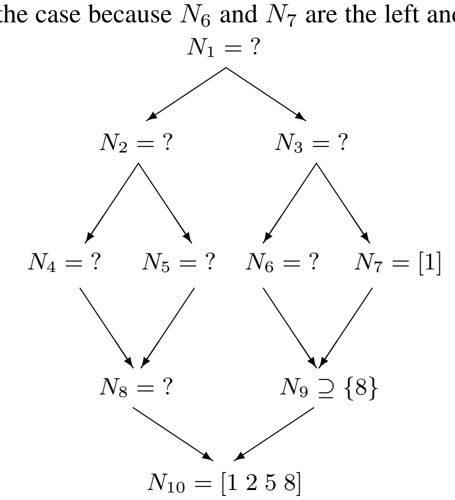


Figure 1.6: A partially unknown MergeSort call graph resulting in the output [1 2 5 8].

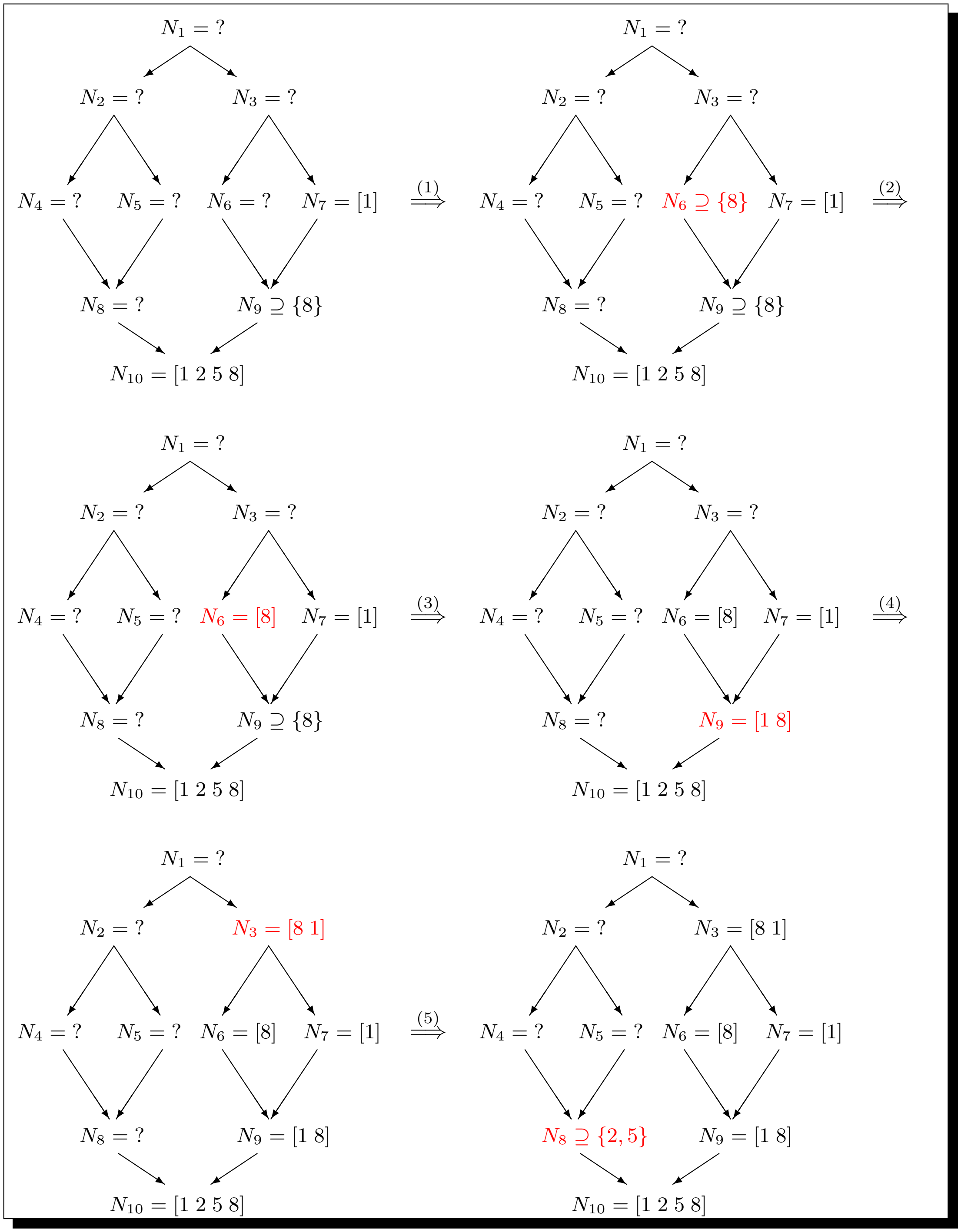


Figure 1.7: First part of a graphical proof solving an instance of the Mergesort puzzle.

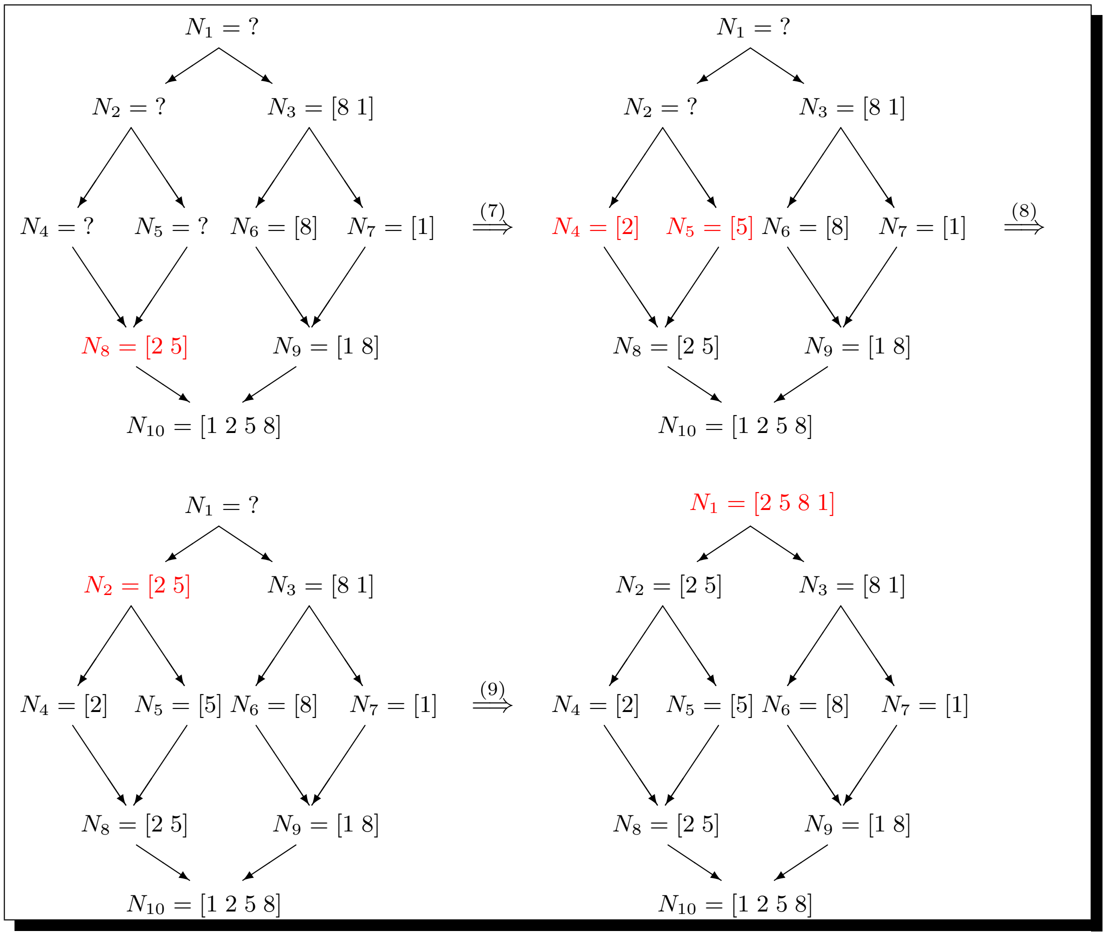


Figure 1.8: Case 1 (out of 2) following the derivation of Figure 1.7.

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.

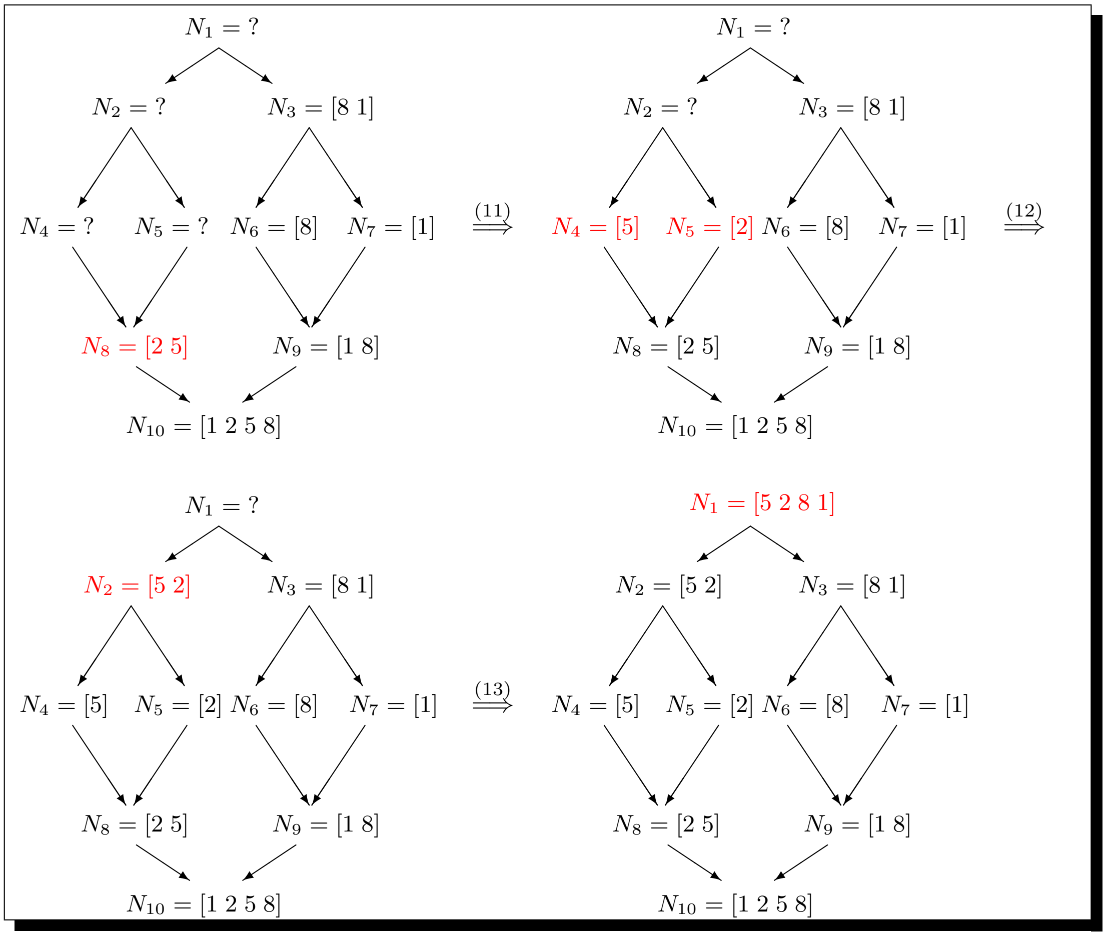


Figure 1.9: Case 2 (out of 2) following the derivation of Figure 1.7.

- Step (6) infers that the N_8 list must be precisely [2 5]. [12] 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 possibilites. 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 concistent 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:

[12]The result of this step does not appear in Figure 1.7 for space reasons, but it is shown as the common starting point of the subsequent case analysis in Figure 1.8 and Figure 1.9.

Case 1 : In that case (Figure 1.8), 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 1.9), 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].

1.7.2 Formalizing the puzzle as an instance of VIVID

There are three steps to obtaining a particular instance of VIVID:

1. Specify an attribute structure $\mathcal{A}$.
2. Specify a vocabulary Σ.
3. Specify an interpretation of the relation symbols of Σ into $\mathcal{A}$ as discussed in Section 1.4.

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:

$$\mathcal{A}_M = (\mathit{id} : \mathit{Node}, \mathit{children} : \mathit{Node}^*, \mathit{data} : Z^*; \{R_1, R_2, R_3, R_4, R_5, R_6, R_7\} \cup \{R_L \mid L \in Z^*\})$$

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

- $R_1 \subseteq \mathit{Node}^* \times \mathit{Node} \times \mathit{Node}$, with
$$R_1([n_1 \ \ldots \ n_k], n, n') \Leftrightarrow \{n, n'\} \subseteq \{n_1, \ldots, n_k\}.$$
- $R_2 \subseteq \mathit{Node} \times \mathit{Node}^* \times \mathit{Node}^*$, with
$$R_2(n, [n_1 \ \ldots \ n_k], [n'_1 \ \ldots \ n'_m]) \Leftrightarrow n \in \{n_1, \ldots, n_k\} \cap \{n'_1, \ldots, n'_m\}.$$
- $R_3 \subseteq Z^* \times Z^* \times Z^*$, with
$$R_3([x_1 \ \ldots \ x_k], [y_1 \ \ldots \ y_n], [z_1 \ \ldots \ z_m]) \Leftrightarrow [x_1 \ \ldots \ x_k] = [y_1 \ \ldots \ y_n \ z_1 \ \ldots \ z_m],$$
i.e., iff $[x_1 \ \ldots \ x_k]$ is the concatenation of $[y_1 \ \ldots \ y_n]$ and $[z_1 \ \ldots \ z_m]$.
- $R_4 \subseteq Z^* \times Z^*$, with
$$R_4([x_1 \ \ldots \ x_k], [y_1 \ \ldots \ y_n]) \Leftrightarrow n \in \{k, k+1\}.$$
- $R_5 \subseteq Z^*$, with
$$R_5([x_1 \ \ldots \ x_k]) \Leftrightarrow x_i \leq x_{i+1} \text{ for } i = 1, \ldots, k-1,$$
i.e., iff $[x_1 \ \ldots \ x_k]$ is sorted.

- $R_6 \subseteq Z^* \times Z^* \times Z^*$, with

$$R_6([x_1 \ \ldots \ x_k], [y_1 \ \ldots \ y_n], [z_1 \ \ldots \ z_m]) \Leftrightarrow \{x_1, \ldots, x_k\} = \{y_1, \ldots, y_n\} \cup \{z_1, \ldots, z_m\}.$$

- $R_7 \subseteq Z^* \times Z^* \times Z^*$, with

$$R_7([x_1 \ \ldots \ x_k], [y_1 \ \ldots \ y_n], [z_1 \ \ldots \ z_m]) \Leftrightarrow k = n + m.$$

- $R_L \subseteq Z^*$, with

$$R_{[x_1 \ \ldots \ x_k]}([y_1 \ \ldots \ y_n]) \Leftrightarrow [x_1 \ \ldots \ x_k] = [y_1 \ \ldots \ y_n].$$

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 1.11 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 $\texttt{val}_L$. 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 1.10.

More intuitive explanations follow:

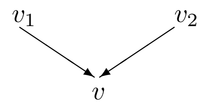
- $\texttt{peak}(v_1, v_2, v_3)$ holds iff nodes v_2 and v_3 are both children of v_1:

$$\begin{tikzcd} & v_1 \arrow[dl] \arrow[dr] & \\ v_2 & \cdots & v_3 \end{tikzcd}$$

- $\texttt{valley}(v_1, v_2, v_3)$ holds iff v_2 and v_3 are both parents of v_1:

Symbol	Arity	Realization	Profile
`peak`	3	R_1	$[(\textit{children}, 1), (\textit{id}, 2), (\textit{id}, 3)]$
`valley`	3	R_2	$[(\textit{id}, 1), (\textit{children}, 2), (\textit{children}, 3)]$
`append`	3	R_3	$[(\textit{data}, 1), (\textit{data}, 2), (\textit{data}, 3)]$
`halves`	2	R_4	$[(\textit{data}, 1), (\textit{data}, 2)]$
`sorted`	1	R_5	$[(\textit{data}, 1)]$
`union`	3	R_6	$[(\textit{data}, 1), (\textit{data}, 2), (\textit{data}, 3)]$
`sum`	3	R_7	$[(\textit{data}, 1), (\textit{data}, 2), (\textit{data}, 3)]$
$\texttt{val}_L$	1	R_L	$[(\textit{data}, 1)]$

Figure 1.10: The interpretation of the Mergesort puzzle vocabulary.



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

```
datatype Nat = zero | succ of Nat;

datatype Node = node of Nat;

fun member(x,L) = List.exists (fn y => x = y) L;

fun subset(L1,L2) = List.all (fn x => member(x,L2)) L1;

fun R1(L,n1,n2) = member(n1,L) andalso member(n2,L);

fun R2(n,L1,L2) = member(n,L1) andalso member(n,L2);

fun R3(L1,L2,L3) = L1 = L2@L3;

fun R4(L1,L2) = let val len1 = length L1
                    val len2 = length L2
                in
                   len2 = len1 orelse len2 = len1 + 1
                end;

fun R5([]) = true
  | R5(x::L) = R5(L) andalso List.all (fn y => x <= y) L;

fun R6(L1,L2,L3) = let val L = L2@L3
                   in
                      subset(L1,L) andalso subset(L,L1)
                   end;

fun R7(L1,L2,L3) = length(L1) = length(L2) + length(L3);
```

Figure 1.11: SML code implementing the attribute structure of the MergeSort puzzle.

1.7.3 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.

$$
\begin{array}{cl}
\forall\, v_1, v_2, v_3 \,.\, \texttt{peak}(v_1, v_2, v_3) \Rightarrow \texttt{halves}(v_2, v_3) & \textit{halves-axiom} \\
\forall\, v_1, v_2, v_3 \,.\, \texttt{valley}(v_1, v_2, v_3) \Rightarrow \texttt{sorted}(v_1) \wedge \texttt{sorted}(v_2) \wedge \texttt{sorted}(v_3) & \textit{sorted-axiom} \\
\forall\, v_1, v_2, v_3 \,.\, \texttt{valley}(v_1, v_2, v_3) \vee \texttt{peak}(v_1, v_2, v_3) \Rightarrow \texttt{union}(v_1, v_2, v_3) & \textit{union-axiom} \\
\forall\, v_1, v_2, v_3 \,.\, \texttt{peak}(v_1, v_2, v_3) \Rightarrow \texttt{append}(v_1, v_2, v_3) & \textit{append-axiom} \\
\forall\, v_1, v_2, v_3 \,.\, \texttt{valley}(v_1, v_2, v_3) \Rightarrow \texttt{sum}(v_1, v_2, v_3) & \textit{sum-axiom}
\end{array}
$$

Now let $\textit{node}_1, \ldots, \textit{node}_{10}$ be ten nodes from the universe of all nodes, *Node*. In combination with the attribute structure $\mathcal{A}_M$, these ten nodes constitute a system. The diagrams shown in Figure 1.7, Figure 1.8 and Figure 1.9 depict specific named states of this system. Consider, for instance, the starting diagram, at the upper left corner of Figure 1.7. This represents a named state $(\sigma; \rho)$, where the partial constant assignment ρ is

$$N_1 \mapsto \textit{node}_1, N_2 \mapsto \textit{node}_2, \ldots, N_{10} \mapsto \textit{node}_{10} \tag{1.109}$$

(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):

$$
\begin{array}{rcl}
\textit{children}(\textit{node}_1) & = & [\textit{node}_2\ \ \textit{node}_3] \\
 & \vdots & \\
\textit{children}(\textit{node}_5) & = & [\textit{node}_8] \\
 & \vdots & \\
\textit{children}(\textit{node}_{10}) & = & []
\end{array}
$$

and

$$
\begin{array}{rcl}
\textit{data}(\textit{node}_1) & = & \{[], [1], [2], [5], [8], [1\ 2], [1\ 5], \ldots, [8\ 5\ 1], \ldots, [1\ 2\ 5\ 8]\} \\
 & \vdots & \\
\textit{data}(\textit{node}_9) & = & \{[8], [1\ 8], [8\ 1], [2\ 8], \ldots, [5\ 8\ 1], [2\ 5\ 1\ 8], \ldots\} \\
 & \vdots & \\
\textit{data}(\textit{node}_{10}) & = & \{[1\ 2\ 5\ 8]\}.
\end{array}
$$

Observe the equation for $\textit{data}(\textit{node}_1)$. At this point we do not know anything about what list appears at $\textit{node}_1$ (a complete lack of knowledge signified by the inscription $N_1 = ?$), so the data field of $\textit{node}_1$ 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* ascription maps every "questionmark 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 $\textit{node}_9$ (the node labeled by N_9) 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,

τ_2 **by thinning with** *union-axiom*;
τ_3 **by thinning with** *halves-axiom*;
τ_4 **by thinning with** *union-axiom*, *sorted-axiom*;
τ_5 **by thinning with** *append-axiom*;
τ_6 **by thinning with** *union-axiom*;
τ_7 **by thinning with** *sum-axiom*, *sorted-axiom*;
cases from *union-axiom*, *halves-axiom***:**
  $\tau_8 \rightarrow \tau_9$ **by thinning with** *append-axiom*;
    τ_{10} **by thinning with** *append-axiom*;
    observe $\mathtt{val}(N_1, [2\ 5\ 8\ 1]) \vee \mathtt{val}(N_1, [5\ 2\ 8\ 1])$
  $\tau_{12} \rightarrow \tau_{13}$ **by thinning with** *append-axiom*;
    τ_{14} **by thinning with** *append-axiom*;
    observe $\mathtt{val}(N_1, [2\ 5\ 8\ 1]) \vee \mathtt{val}(N_1, [5\ 2\ 8\ 1])$

Figure 1.12: Formal VIVID proof solving the Mergesort puzzle of Section 1.7.1.

the value of *data* for $node_{10}$ is completely determined—the singleton $\{[1\ 2\ 5\ 8]\}$. [13] 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 ϕ 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 ψ that proceeds in the reverse direction, rendering system states graphically. As with customary parsing and unparsing, we have

$$\psi(\phi(d)) = d \text{ and } \phi(\psi(\sigma)) = \sigma \tag{1.110}$$

for all diagrams d and system states σ, where the first identity is understood to obtain up to topological equivalence. [14] 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 ϕ and ψ can be computed efficiently—in low polynomial time—using standard graph-theoretic algorithms.

Finally, Figure 1.12 shows the formal VIVID proof that solves the Mergesort puzzle discussed in Section 1.7.1. We conclude with a detailed analysis of this proof.

First, we need a simple lemma:

$$\forall\, v_1, v_2, v_3 \,.\, \mathtt{valley}(v_1, v_2, v_3) \Rightarrow \mathtt{union}(v_1, v_2, v_3) \wedge \mathtt{sum}(v_1, v_2, v_3) \quad [\textit{lemma}]$$

[13]These are unnecessarily coarse approximations. We could leverage our knowledge of the domain to further cut down the possibililities 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.

[14]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" (Quine 1969) comes to mind). Nevertheless, there are many cases where we can formulate rigorous necessary and sufficient conditions for two diagrams to be considered identical, using topological or other extensional notions.

This can be derived from our five axioms in a few lines of VIVID, by some elementary sentential reasoning; we leave the derivation to the reader.

Next, let $\sigma_1, \ldots, \sigma_6$ be the system states corresponding to the six diagrams that appear in Figure 1.9 starting from the top left corner and proceeding clockwise, so that σ_i represents the graph to the left of the arrow indicating the i^{th} step. Likewise, let $\sigma_7, \ldots, \sigma_{10}$ and $\sigma_{11}, \ldots, \sigma_{14}$ be the states corresponding to the diagrams of Figure 1.8 and Figure 1.9, respectively. For any $i = 1, \ldots, 14$, we write τ_i to denote the named state $(\sigma_i; \rho)$, where ρ is the constant assignment (1.109).

Recalling that composition is right-associative, we see that the proof in Figure 1.12 is a sentential proof D, as it is of the form

$$D = \Delta_1; \cdots; \Delta_6; D',$$

i.e., a composition of six diagrammatic steps $\Delta_1, \ldots, \Delta_6$ followed by a sentential deduction D' of the form

$$\textbf{cases from } F_1, \ldots, F_k\textbf{:}\ (\sigma_1; \rho_1) \ \rightarrow\ D_1 \mid \cdots \mid (\sigma_n; \rho_n) \ \rightarrow\ D_n,$$

a diagrammatic-to-sentential case analysis. The starting point for the proof is the context

$$\gamma_1 = (\beta_1; \tau_1), \tag{1.111}$$

where β_1 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 why the first step Δ_1, the diagrammatic inference

$$\tau_2 \ \textbf{by thinning with}\ \textit{union-axiom}$$

succeeds. According to the semantics of thinning (Figure 1.3), this step will be valid provided that

$$\tau_1 \Vdash\!\!\vdash_{\{\textit{union-axiom}\}} \tau_2,$$

i.e., provided that τ_1 entails τ_2 with respect to *union-axiom*. This means that every alternative way of extending τ_1 w.r.t. τ_2 must falsify *union-axiom* (for an arbitrary variable assignment). More precisely, it must be the case that for every named state $\tau = (\sigma; \rho)$ such that $\mathit{Alt}(\tau_1, \tau_2, \tau)$ we have

$$I_{(\sigma;\,\rho)/\chi}(\textit{union-axiom}) = \textbf{false} \tag{1.112}$$

for all χ. Pick any such τ. Since τ_1 and τ_2 share the same constant assignment ρ, the only way τ can be an alternative extension of τ_1 w.r.t. τ_2 is if we have $\mathit{Alt}(\sigma_1, \sigma_2, \sigma)$ (Definition 10). The only state σ that qualifies as such an alternative is the one that is identical to σ_1 except that the *data* ascription maps node_6 to the set of all lists in $\mathcal{L}$ that do *not* contain 8. It is easy to see that (1.112) holds in that state. Indeed, consider an arbitrary χ. By (1.5), *union-axiom* will be false in $(\sigma; \rho)$ and χ if there are some nodes node_{i_1}, node_{i_2}, and node_{i_3} such that

$$I_{(\sigma;\,\rho)/\chi[v_1 \mapsto \mathit{node}_{i_1}, v_2 \mapsto \mathit{node}_{i_2}, v_3 \mapsto \mathit{node}_{i_3}]}(\texttt{valley}(v_1, v_2, v_3) \vee \texttt{peak}(v_1, v_2, v_3) \Rightarrow \texttt{union}(v_1, v_2, v_3)) = \textbf{false}.$$

Let these three nodes be node_9, node_6, and node_7, respectively (i.e., the nodes labeled by N_9, N_6 and N_7). For these nodes we clearly have:

$$I_{(\sigma;\,\rho)/\chi[v_1 \mapsto \mathit{node}_9, v_2 \mapsto \mathit{node}_6, v_3 \mapsto \mathit{node}_7]}(\texttt{valley}(v_1, v_2, v_3) \vee \texttt{peak}(v_1, v_2, v_3)) = \textbf{true}$$

(since the nodes form a valley) and yet

$$I_{(\sigma;\,\rho)/\chi[v_1 \mapsto \mathit{node}_9, v_2 \mapsto \mathit{node}_6, v_3 \mapsto \mathit{node}_7]}(\texttt{union}(v_1, v_2, v_3)) = \textbf{false}. \tag{1.113}$$

(1.113) holds because for *every* list L in the *data* field of *node*$_9$ in σ and for every list L' in the *data* field of *node*$_6$ in σ and every list L'' in the *data* field of *node*$_7$ in σ, we have

$$\neg R_6(L, L', L''),$$

the reason being that every such L contains 8 but no such L'' contains 8 (because *data*(*node*$_7$) in σ contains only one list value, [1]) and no such L' contains 8 (by virtue of σ being an alternative extension of σ_1 w.r.t. σ_2).

It is important to note that in practice these three nodes would be discovered automatically by exhaustive search. Specifically, the system would evaluate the formula *union-axiom* in the named state $(\sigma; \rho)$and an arbitrary variable assignment χ [15] to determine if it comes out **false**. Now a universally quantified formula such as *union-axiom* is evaluated in a given χ by binding the universally quantified variable to successive system objects and recursively evaluating the body in the updated χ. If the body comes out **false** for some system object, the whole formula is deemed **false**. If the body itself is another universally quantified formula then we have more choice points and possible backtracking. In the worst case for the puzzle example, the evaluation of *union-axiom* will need to examine $10^3 = 1000$ difference possible assignments of variables to objects, since the system comprises 10 nodes and the formula has three universally quantified variables. In such a worst-case scenario, the body of *union-axiom* would be evaluated for each of the 1000 node triples. For most of these triples, *union-axiom* would come out **unknown** because there is not enough information to enable a definitive judgment. Consider, for instance, the evaluation of the body of *union-axiom* in the triple

$$\chi[v_1 \mapsto \textit{node}_1, v_2 \mapsto \textit{node}_2, v_3 \mapsto \textit{node}_3].$$

While it is true that *node*$_1$, *node*$_2$, and *node*$_3$ form a peak, we have

$$I_{(\sigma;\rho)/\chi[v_1 \mapsto \textit{node}_1, v_2 \mapsto \textit{node}_2, v_2 \mapsto \textit{node}_3]}(\texttt{union}(v_1, v_2, v_3)) = \textbf{unknown}$$

because, in σ, the realization of `union`, R_6, holds for some list values in the corresponding *data* fields of *node*$_1$, *node*$_2$ and *node*$_3$ and does not hold for others (see (1.3)).

In this particular example we have 10 system objects and the most populous attribute value has 65 elements, so a formula such as $\texttt{union}(v_1, v_2, v_3)$ could, in theory, take up to $65^3 = 274,625$ evaluations to settle. Combined with the 1,000 triple possibilities dictated by three universal quantifiers, we could look at the non-trivial number of $274,625,000$ evaluations. However, in practice atomic formulas such as $\texttt{union}(v_1, v_2, v_3)$ would be settled speedily because for most node assignments we would get some **true** and some **false** values, quickly leading to an **unknown** result. So even the worst case of 1000 different evaluations is not computationally formidable.

Nevertheless, we observe that the user can always improve the efficiency of the proof checking by providing more information in the proof—information that guides the searh in the right direction. For example, we could replace the first step

τ_2 **by thinning with** *union-axiom*

by the following sequence of steps:

specialize *union-axiom* **with** N_9, N_6, N_7;
observe $\texttt{valley}(N_9, N_6, N_7)$;
right-either $\texttt{peak}(N_9, N_6, N_7) \vee \texttt{valley}(N_9, N_6, N_7)$;
modus-ponens $\texttt{peak}(N_9, N_6, N_7) \vee \texttt{valley}(N_9, N_6, N_7) \Rightarrow \texttt{union}(N_9, N_6, N_7)$,
$\texttt{peak}(N_9, N_6, N_7) \vee \texttt{valley}(N_9, N_6, N_7)$;
τ_2 **by thinning with** $\texttt{union}(N_9, N_6, N_7)$

[15] This is legitimate by virtue of Lemma 2.

Here we focus directly on the three nodes of interest by citing $\mathtt{union}(N_9, N_6, N_7)$ as the justification of the thinning step, instead of citing the universally quantified *union-axiom*. By eliminating the three universal quantifiers, we avert the need to evaluate the body of *union-axiom* over all possible triples of nodes. The tradeoff is a typical manifestation of the usual tension between brevity and efficiency: a very brief proof takes large steps whose verification can be difficult because it requires search; whereas a detailed proof takes small steps that are easy to check because they involve little or no search. We can always buy efficiency at the expense of consiceness.

Returning to the proof, let us examine the second step:

τ_3 **by thinning with** *halves-axiom*.

As with the previous application of thinning, this step is valid only if

$$\tau_2 \Vvdash_{\{\textit{halves-axiom}\}} \tau_3,$$

meaning that any named state $\tau = (\sigma; \rho)$ that is an alternative extension of τ_2 w.r.t. τ_3 must falsify *halves-axiom*. As before, because the constant assignment does not change, the only way we can have $\mathit{Alt}(\tau_2, \tau_3, \tau)$ is if we have $\mathit{Alt}(\sigma_2, \sigma_3, \sigma)$. And given that in σ_2 the *data* field of $node_6$ contains all and only those lists that contain 8, σ is an alternative extension of σ_2 w.r.t. σ_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 next four thinning steps. We encourage the reader to work through them rigorously.

We come finally to the case analysis, which turns on the claim that from the state σ_7 and on the basis of the lemma, there are only two possible states, σ_8 and σ_{12}. Symbolically,

$$(\sigma_7; \rho) \Vvdash_{\{\textit{lemma}\}} \{(\sigma_8; \rho), (\sigma_{12}; \rho)\}. \tag{1.114}$$

Consulting Definition 11, we see that (1.114) holds iff for every $(\sigma'; \rho')$ such that

$$\mathit{Alt}((\sigma_7; \rho), \{(\sigma_8; \rho), (\sigma_{12}; \rho)\}, (\sigma'; \rho')) \tag{1.115}$$

we have $I_{(\sigma'; \rho')/\chi}(\textit{lemma}) = \textbf{false}$ for all χ. Again, because the constant assignment does not change, (1.115) holds iff

$$\mathit{Alt}(\sigma_7, \{\sigma_8, \sigma_{12}\}, \sigma') \tag{1.116}$$

(by Definition 10).

Now there are two alternative extensions of σ_7 w.r.t. $\{\sigma_8, \sigma_{12}\}$: one, call it σ_A, in which we keep the $\{[2], [5]\}$ value of $node_4$ steady but complement it for $node_5$; while the other, call it σ_B, is one in which we complement the *data* value of $node_4$ in σ_7 and retain the *data* value of $node_5$. The relevant parts of both states can be depicted graphically as follows:

$$\sigma_A: \quad \begin{tikzcd}[column sep=tiny] & N_2 = ? \arrow[dl] \arrow[dr] & \\ N_4 = \{[2], [5]\} \arrow[dr] & & N_5 = \mathcal{L} \setminus \{[2], [5]\} \arrow[dl] \\ & N_8 = [2\ 5] & \end{tikzcd} \qquad \sigma_B: \quad \begin{tikzcd}[column sep=tiny] & N_2 = ? \arrow[dl] \arrow[dr] & \\ N_4 = \mathcal{L} \setminus \{[2], [5]\} \arrow[dr] & & N_5 = \{[2], [5]\} \arrow[dl] \\ & N_8 = [2\ 5] & \end{tikzcd}$$

A routine calculation will confirm that both possibilities falsify the cited lemma.

1.8 Related Work

We have derived much inspiration from the seminal work of Barwise, Etchemendy, and others on Hyperproof (Barwise and Etchemendy 1995b). One of the chief contributions of Hyperproof was its emphasis on incomplete information and its ability to reason about ambiguous (partially determined) situations. These choices are not only pedagogically sound, since there are many types of reasoning problems[16] 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, since 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.[17]

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 limited and ad hoc. For instance, although a diagram can signify that the size of a block is unknown, it has no way of indicating that it is, say, large or medium but not small. By contrast, VIVID's mechanism for handling incomplete diagrammatic information via arbitrary sets of values is completely general.

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.a). 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 using vertical lines to represent lexical scope in programming languages (instead of the usual begin-end pairs or curly braces) would scale to realistic programs.

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 the field of 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 neat induction proofs on derivations (witness our soundness proof). 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.[18]

5. Because it is based on DPLs, VIVID could be extended 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 would be guaranteed by the formal semantics of the language (see (Arkoudas n.d.c) for an example of how such extensions are actually performed). It is not at all clear how Hyperproof could be made programmable, let alone in a way that would guarantee soundness.

[16]E.g., in logical and analytical reasoning problems of standardized tests such as GRE or LSAT.

[17]As Konolige and Meyers (Myers and Konolige 1995) put it:

> When generating maps from perceptual input, noise or faulty sensors may both cause objects of interest to go undetected and leave analogical relations only partially determined.

[18]For instance, very efficient proof-simplification algorithms that were developed for $\mathcal{NDL}$ (Arkoudas n.d.b) were made possible—and proven sound—owing to the formal operational semantics of the language. The same ideas could be incorporated into VIVID, resulting in general principles and procedures for eliminating redundant reasoning from diagrammatic proofs.

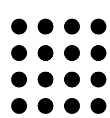
6. Hyperproof is proprietary; VIVID is in the public domain. The difference is not without practical ramifications. The open design of VIVID enables highly modular implementations because it exposes a sharp separation between the purely graphical tasks of diagram parsing and unparsing on one hand and the system's syntax, semantics, and underlying diagrammatic inference procedures on the other. The latter are fixed once and for all and proven sound. All one needs to do in order to implement a specific instance of VIVID is fix a class of diagrams and provide a diagram parser (compiling diagrams into system states) and unparser (rendering system states graphically). Hyperproof is much more of a monolithic black box, and any attempt by third parties to build Hyperproof-like systems for other domains would have to resort to reverse engineering.

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 of 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 specific, 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 diagrammating proofs of certain types of arithmetic theorems. The system is designed to reason exclusively aboyt natural numbers, and specifically with universally quantified identities of the form $\forall \cdots . s = t$, 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

$$\sum_{i=1}^{n} 2i - 1 = n^2 \qquad (1.117)$$

Diagrammatic proofs are only given for particular *instances* of the theorem, e.g., for (1.117) 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 4×4 square matrix of dots:



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 metatheoretic framework. This is probably the most problematic step of the process, as the problem is undecidable in general. We are thus faced with the somewhat odd consequence that even though DIAMOND is only a proof checker and not a proof finder, it might nevertheless still 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.

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 &. Its authors report having used the system 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. In their view, a diagram represents a trail of the objects that are involved in the proof, along with key properties of and relation among such objects. This is an interesting view of diagrams, but it differs from the (more rigorous) sense in which diagrams are used in systems such as Hyperproof or VIVID, where diagrams are essentially used 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 tesselation 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 some 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 rather 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 symbolic information.

1.9 Conclusions

A cursory reading of this paper might leave one asking: "So where are the diagrams? All I see are sets and lists and functions and so on—the usual sentential stuff." Indeed, as Greaves (Greaves 2002) correctly states:

> Diagrammatic representations can be recognized by the extent to which the geometric properties of the components of the representation are relevant to their interpretation, and the ways in which these properties impact the reasoning methods which are licenced by the overall theory.

But our theory revolves around attribute structures, system states, etc., and has ostensibly little to do with "geometric properties" of any kind.

There is nothing odd about that. Our theory is a logical analysis of the computational and information-theoretic aspects of certain types of diagrammatic reasoning. It is not itself a piece of diagrammatic reasoning, nor does it need to be. A mathematical analysis of visual inference does not itself need to be visual any more than a mathematical analysis of acoustics needs to be musical, or any more than a mathematical analysis of heat needs to be hot.

Still, one might wonder whether any representation of diagrams by set-theoretic structures is not bound to lose something of the essentially pictorial nature of diagrams. Perhaps, but the issue is rather orthogonal to our concerns. Our analysis is mostly motivated by engineering concerns and is therefore given with a view to building robust, efficient, usable systems that permit perspicuous heterogeneous proofs combining diagrammatic and sentential reasoning.

In summary, we have introduced VIVID, a family of denotational proof languages (DPLs) that combine sentential and diagrammatic reasoning in a Fitch-style natural deduction framework. VIVID is based on the notion of attribute systems, and on the use of Kleene's strong three-valued logic to interpret first-order signatures into attribute structures. To obtain a particular instance of VIVID, we need only specify an attribute structure, a signature, and an interpretation of the signature into the structure.

We have not discussed how diagrams would be concretely represented within the proof text. That is an interface issue, not an issue of 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 pop up, 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 drawn depends on the specific application domain at hand; it is completely separate from all other aspects of the language. This modularity could be put to good use, e.g., an implementation of VIVID could be designed as an SML functor (Paulson 1996) that will take an attribute strucure $\mathcal{A}$; the interpretation of a signature Σ into $\mathcal{A}$; and a drawing module that can draw an arbitrary $\mathcal{A}$-system; and will output a parser and an interpreter, i.e., a proof checker for the instantiated language.

Introducing names brings up another possibility. As it stands, an implementation of VIVID would be a type-α DPL, i.e., a proof checker: it would accept a proof combining sentential and diagrammatic steps and would either pronounce it sound or else point out a reasoning error. If we introduce unrestricted naming and computation, we can make these into type-ω DPLs (Arkoudas n.d.c, Arvizo n.d.), capable not only of proof checking but of arbitrary proof search as well. It would be very interesting to see what types of methods can be written in such a setting for the purpose of automating diagrammatic inference, and exactly what type of formal soundness guarantee we might be able to provide.

Another important issue is efficiency. Depending on the system we are working with, we may need exponential time in the size of the attributes to check whether an application of a rule such as thinning is valid. This is due solely to the size of the attributes and is orthogonal to how "large" are the steps taken by the user. Even if the user takes a very small step, say to exclude one possible value from a set thereof, we may still need to explore exponentially many subsets. Two possibilities for ameliorating this issue are: (a) representing sets of attribute values by binary decision diagrams (BDDs) (Bryant 1992), and (b) symbolic evaluation. For (a), it is hoped that a compact representation of the relevant subsets might speed up rule checking. There are standard techniques for representing an arbitrary subset S' of any finite set S by a BDD, basically by encoding the characteristic function of S' as a Boolean function (Huth and Ryan 2000). With symbolic evaluation, we may be able to prune very large parts of the search tree if we incorporate a modest degree of domain knowledge into the search process. For instance, if we determine that a time (h_1, m_1) of a clock c_1 is not ahead of some clock c_2, there is no point in trying other possible times (h'_1, m'_1) of c_1 if $h'_1 < h_1$ or if $h'_1 = h_1$ and $m'_1 < m_1$. Sophisticated techniques for performing symbolic predicate evaluation (similar to the symbolic evaluation methods in model checking (Clarke et al. 1999)) could have a significant payoff.

Bibliography

Agusti, J., Puigsegur, J. and Robertson, D. S.: 1998, A visual syntax for logic and logic programming, *Journal of Visual Languages and Computing* **9**(4), 399–427.

Anderson, M. and McCartney, R.: 2003, Diagram processing: Computing with diagrams, *Artificial Intelligence* **145**(1–2), 181–226.

Arkoudas, K.: 2000, *Denotational Proof Languages*, PhD thesis, MIT, Department of Computer Science, Cambridge, USA.

Arkoudas, K.: n.d.a, Type-α DPLs. MIT AI Memo 2001-25.

Arkoudas, K.: n.d.b, Simplifying proofs in Fitch-style natural deduction systems. Accepted for publication in the Journal of Automated Reasoning, December 2004.

Arkoudas, K.: n.d.c, Type-ω DPLs. MIT AI Memo 2001-27.

Arvizo, T.: n.d., A virtual machine for a type-ω denotational proof language. Masters thesis, MIT, June 2002.

Barker-Plummer, D. and Bailin, S. C.: 1992, Proofs and pictures: Proving the diamond lemma with the GROVER theorem proving system, *Working notes of the AAAI Spring Symposium on Reasoning with Diagrammatic Representations*, American Association for Artificial Intelligence, Cambridge, MA.

Barwise, J. and Etchemendy, J.: 1995a, Heterogeneous logic, *in* J. Glasgow, N. Narayanan and N. H. Chandrasekaran (eds), *Diagrammatic Reasoning*, MIT Press, Cambidge, USA, pp. 211–234.

Barwise, J. and Etchemendy, J.: 1995b, *Hyperproof: for Macintosh*, CSLI Publications.

Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L. and Schnoebelen, Ph.: 2001, *Systems and Software Verification. Model-Checking Techniques and Tools*, Springer.

Bier, E. A., Stone, M. C., Pier, K., Buxton, W. and DeRose, T. D.: 1993, Toolglass and magic lenses: The see-through interface, *Computer Graphics* **27**(Annual Conference Series), 73–80.

Bryant, R. E.: 1992, Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams, *ACM Computing Surveys* **24**(3), 293–318.

Chang, S.-K. (ed.): 1990, *Principles of Visual Programming Systems*, Prentice Hall, New York.

Clarke, E. M., Grumberg, O. and Peled, D. A.: 1999, *Model checking*, MIT Press.

Cormen, T., Leiserson, C. and Rivest, R.: 1990, *Introduction to Algorithms*, MIT Press.

Eggleston, H. G.: 1969, *Convexity*, Cambridge University Press.

Englebretsen, G.: 1992, Linear diagrams for syllogisms (with relationals), *Notre Dame Journal of Formal Logic* **33**(1), 37–69.

Etherington, D. W., Borgida, A., Brachman, R. J. and Kautz, H. A.: 1989, Vivid knowledge and tractable reasoning: preliminary report, *Proceedings of IJCAI-89, 10th International Joint Conference on Artificial Intelligence*, Detroit, US, pp. 1146–1152.

Euler, L.: 1768, Lettres à une Princesse d'Allemagne, *l'Academie Imperiale des Sciences* .

Fagin, R., Mendeizon, A. and Ullman, J.: 1982, A simplified universal relation assumption and its properties, *ACM Transactions on Database Systems* **7**(3), 343–360.

Greaves, M.: 2002, *The Philosophical Status of Diagrams*, CSLI Publications, Stanford, California.

Grigni, M., Papadias, D. and Papadimitriou, C. H.: 1995, Topological inference, *Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal*, pp. 901–905.

Hammer, E. M.: 1995, *Logic and Visual Information*, CSLI Publications, Stanford, California.

Harel, D.: 1988, On visual formalisms., *Commun. ACM* **31**(5), 514–530.

Hirakawa, M., Tanaka, M. and Ichikawa, T.: 1990, An iconic programming system, HI–VISUAL, *IEEE Transactions on Software Engineering* **16**(10), 1178–1184.

Huth, M. R. A. and Ryan, M. D.: 2000, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, Cambridge, England.

Jamnik, M.: 2001, *Mathematical Reasoning With Diagrams*, CSLI Publications, Stanford, California.

Johnson, S. D., Barwise, J. and Allwein, G.: 1996, Towards the rigorous use of diagrams in reasoning about hardware, *in* G. Allwein and J. Barwise (eds), *Logical Reasoning with Diagrams*, Oxford University Press, pp. 201–223.

Kahn, G.: 1987, Natural semantics, *Proceedings of Theoretical Aspects of Computer Science*, Passau, Germany.

Kleene, S.: 1952, *Introduction to metamathematics*, North-Holland, Amsterdam.

Lemon, O.: 2002, Comparing the Efficacy of Visual Languages, *in* D. Barker-Plummer, D. I. Beaver, J. van Benthem and P. S. di Luzio (eds), *Words, Proofs, and Diagrams*, CSLI Publications, Stanford, California, pp. 47–69.

Lemon, O. and Pratt, I.: 1997, Spatial Logic and the Complexity of Diagrammatic Reasoning, *Machine Graphics and Vision* **6**(1), 89–109.

Levesque, H. J.: 1989, Making believers out of computers, *in* J. Mylopoulos and M. L. Brodie (eds), *Artificial Intelligence & Databases*, Kaufmann Publishers, INC., San Mateo, CA, pp. 69–82.

Manbelbrot, B.: 1982, *The Fractal Geometry of Nature*, W. H. FFreeman and Company.

Meinke, K. and Tucker, J. V.: 1992, Universal algebra, *in* S. Abramsky, D. M. Gabbay and T. S. E. Maibaum (eds), *Handbook of Logic in Computer Science: Background - Mathematical Structures (Volume 1)*, Clarendon Press, Oxford, pp. 189–411.

Myers, K. and Konolige, K.: 1995, Reasoning with analogical representations, *in* J. Glasgow, N. Narayanan and N. H. Chandrasekaran (eds), *Diagrammatic Reasoning*, MIT Press, Cambidge, USA, pp. 273–301.

Myers, K. L.: 1994, Hybrid Reasoning Using Universal Attachment, *Artificial Intelligence* **67**, 329–375.

Ogawa, T. and Tanaka, J.: 2000, CafePie: A Visual Programming System for CafeOBJ, *Cafe: An Approach to Industrial Strength Algebraic Formal Methods*, Elsevier Science, pp. 145–160.

Paulson, L. C.: 1996, *ML for the working programmer*, 2nd edn, Cambridge University Press, Cambridge, England.

Peirce, C.: 1960, *The collected papers of C. S. Peirce*, Harvard University Press.

Pierce, B. C.: 1991, *Basic Category Theory for Computer Scientists*, Foundations of Computing, MIT Press, Cambridge, Massachusetts.

Plotkin, G. D.: 1981, A structural approach to operational semantics, *Research Report DAIMI FN-19*, Computer Science Department, Aarhus University, Aarhus, Denmark.

Quine, W. V. O.: 1969, Speaking of objects, *Ontological Relativity and Other Essays*, Columbia University Press, New York.

Reynolds, J. C.: 1998, *Theories of Programming Languages*, Cambridge University Press.

Rumbaugh, J., Jacobson, I. and Booch, G.: 1999, *The Unified Modeling Language Reference Manual*, Addison-Wesley.

Russell, B.: 1923, Vagueness, *Australasian Journal of Philosophy and Psychology* **1**, 84–92.

Shimojima, A.: 1996, Operational constraints in diagrammatic reasoning, *in* G. Allwein and J. Barwise (eds), *Logical Reasoning with Diagrams*, Oxford University Press, pp. 27–48.

Sloman, A.: 1971, Interactions between philosophy and AI: The role of intuition and non-logical reasoning in intelligence, *Proceedings of the Second International Joint Conference on Artificial Intelligence*.

Sober, E.: 1976, Mental representations, *Synthese* **33**, 101–148.

Stenning, K. and Lemon, O.: 2001, Aligning logical and psychological perspectives on diagrammatic reasoning, *Thinking with Diagrams*, Kluwer.

Veltman, M.: 1995, Diagrammatica: the Path to Feynman Rules, Vol. 4 of *Cambridge Lecture Notes in Physics*, Cambridge University Press.

Wechler, W.: 1992, *Universal Algebra for Computer Scientists*, Springer-Verlag.