Artificial Intelligence 173 (2009) 1367–1405

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Vivid: A framework for heterogeneous problem solving ☆

Konstantine Arkoudas, Selmer Bringsjord *

Rensselaer AI & Reasoning (RAIR) Lab, Department of Cognitive Science, Department of Computer Science, Rensselaer Polytechnic Institute (RPI), Troy, NY 12180, USA

ARTICLE INFO

Article history:
Received 30 June 2008
Received in revised form 1 June 2009
Accepted 9 June 2009
Available online 13 June 2009

Keywords:
Vivid
Heterogeneous reasoning
Problem solving
Diagrams
DPLs
Assumption bases
Named system states
Worlds
3-valued logic

ABSTRACT

We introduce Vivid, a domain-independent framework for mechanized heterogeneous reasoning that combines diagrammatic and symbolic representation and inference. The framework is presented in the form of a family of denotational proof languages (DPLs). We present novel formal structures, called *named system states*, that are specifically designed for modeling potentially underdetermined diagrams. These structures allow us to deal with incomplete information, a pervasive feature of heterogeneous problem solving. We introduce a notion of attribute interpretations that enables us to interpret first-order relational signatures into named system states, and develop a formal semantic framework based on 3-valued logic. We extend the assumption-base semantics of DPLs to accommodate diagrammatic reasoning by introducing general inference mechanisms for the valid extraction of information from diagrams, and for the incorporation of sentential information into diagrams. A rigorous big-step operational semantics is given, on the basis of which we prove that the framework is sound. We present examples of particular instances of Vivid in order to solve a series of problems, and discuss related work.

1. Introduction

Diagrams have been recognized as valuable representational and reasoning tools at least since the days of Euclid. They are used extensively in a very wide range of fields. To note just a few examples, witness: free-body, energy-level and Feynman diagrams in physics [60]; arrow diagrams in algebra and category theory [44]; Euler and Venn diagrams in elementary set theory and logic; 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 [32]; UML diagrams in software design [47]; higraphs in specification [25]; visual programming languages [15] and visual logic and specification languages [1,27,42]; transition graphs in model checking [11]; ER-diagrams and hypergraphs in databases [21]; semantic (as well as neural and belief) networks in AI [50]; icons and other pictorial devices in graphical user interfaces (GUIs) and information visualization [12,39,59,63]; and so on. Given such a list, and the power of diagrams that it suggests, it seems reasonable to hold that if the capability of computers to work with diagrams intelligently can be further increased, human reasoning and problem solving will be facilitated. The framework presented here, Vivid, is intended to purchase some of that increase.

The representational power of diagrams stems primarily from the fact that they can have structural correspondences with the objects or situations they represent—they are *analogical representations* in the terminology of Sloman [54], or *homomorphic representations* in the terminology of Barwise and Etchemendy [7]; also see Hayes [26]. To put it more plainly,

☆ This work was made possible by grants received from DARPA and DTO. We are indebted to Ron Brachman, David Musser, Martin Rinard, and to three anonymous referees for insightful comments, objections, and suggestions.

* Corresponding author.
E-mail addresses: arkouk@rpi.edu (K. Arkoudas), selmer@rpi.edu (S. Bringsjord).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2009.06.002

a diagram *resembles*—or at any rate *should* resemble—what the diagram depicts, in contrast to sentential descriptions.[1] This was noticed at least as far back as the 19th century, when Peirce observed that a diagram is "naturally analogous to the thing represented" [43, p. 316].

Consider, for instance, the task of describing a human face. We could perhaps describe the face with a collection of English sentences, or with a set of sentences in some formal language. But such a description is likely to be long and complicated, and not particularly illuminating.[2] A drawing or a picture of the face, on the other hand, will be much more perspicuous, as well as significantly more compact than most sentential representations. Of course, some diagrams are better than others. A talented artist will produce a drawing that is a much more accurate depiction than the scrawlings of a child. A digital picture will be even more accurate.[3] So, as Hammer [24] observes, being an analogical or homomorphic representation is not a distinguishing feature of diagrams in general, but rather a distinguishing feature of *good* diagrams.

The utility of (good) diagrams is often thought to derive from the fact that diagrams are two-dimensional objects, and therefore spatial relationships on the plane can directly reflect analogous relationships in the underlying domain, an observation made a while back by Russell [49]. 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 doubtless 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 [57] 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.

However, two-dimensionality by itself is neither a necessary nor a sufficient condition for being a diagram. For instance, as Hammer [24] points out, a representation of a picture by a two-dimensional array of numbers encoded under some encryption scheme does not count as a diagram; there is no structural similarity between the representation and that which is being represented. And, by making sufficiently clever conventions, we can construct analogical one-dimensional diagrams. For example, the following string asserts that the stretch of road between Main Street/35th Street and Main/36th is two-way, whereas that between Main/36th and Main/37th is one-way and proceeds from right to left:

```
Main|35th <==> Main|36th <== Main|37th
```

It bears stressing that diagrams are helpful only when their visual structure is analogical or homomorphic with the semantic structure of the information which they represent. In an era of Powerpoint and multimedia presentations, it is often taken for granted that graphical displays of information are automatically clearer and more intuitive than text, simply by virtue of being "visual." That is emphatically not the case. The reason why Euler circles are efficacious, for instance, is precisely because spatial enclosure is naturally analogous to the subset relation, spatial overlap to set-theoretic intersection, and spatial separation to set-theoretic disjointness [53]. In the absence of such structural similarities, diagrams can quickly degenerate into what Tufte [59, p. 34] calls "chartjunk": cluttered displays of lines, curves, arrows, bars, charts, and the like, that end up obscuring rather than clarifying information.[4] Conversely, a diagram does not have to be visually arresting or elaborate in order to be superior to a sentential representation. It does not even have to be two-dimensional, as we noted above, a point that is borne out by our Main Street example, or by Hammer's example of an one-dimensional diagram

> meant to express the relative distances between the Earth, Moon, and Mars when the Moon is aligned to fall between Earth and Mars:
>
> Earth–Moon———Mars
>
> This diagram is one-dimensional: its syntax can be adequately modeled by sequences of symbols [24, p. 2].

Some might be inclined to criticize such diagrams as inordinately simple and purely structural, hence suffering from insufficient "diagrammaticity," the implication being that only visually elaborate diagrams qualify as truly diagrammatic. The criticism is at odds with the brute reality of ingenious human diagrammatic reasoning and problem solving. For example, consider the well-known example of the seating puzzle of Barwise and Etchemendy [6], which we discuss extensively in Section 8. The diagrams in that puzzle are indeed very simple (one-dimensional, small, and purely ASCII); but they are no less powerful for human reasoners as a result. In fact, their structural nature and simplicity, far from being defects, are positively conducive to their representational power. Structure and simplicity are usually advantages of analogical representations, not disadvantages.

[1] The terms "sentential" and "symbolic" will be used synonymously throughout.

[2] Fractals [37] 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 analogical to the corresponding shapes than other symbolic descriptions.

[3] In the limiting case, the ultimate representation of an object is the object itself.

[4] Peter Norvig provides an amusing but compelling illustration of this point in his Powerpoint version of the Gettysburg address, where he turns "four scores and seven years" into a gratuitous graph: www.norvig.com/Gettysburg/sld005.htm. More information can be found at www.norvig.com/Gettysburg/making.html.

In the next section we discuss some design desiderata. In Section 3 we present the notation that we use throughout this paper. Section 4 introduces the main conceptual tools used to specify and investigate the semantics of Vivid: attribute structures, attribute systems, and attribute system states. In Section 5 we develop the theoretical framework necessary for defining and analyzing Vivid. We introduce the notion of attribute interpretations, which enables us to evaluate first-order formulas with respect to attribute system states in accordance with a 3-valued logic. We also introduce named system states, thereby formalizing the idea that arbitrary names can appear inside diagrams. A number of useful results are listed here that are needed later for our soundness theorem.[5] Section 6 presents and discusses the abstract syntax and formal evaluation semantics of Vivid; our main soundness result also appears here. Section 7 discusses implementation issues. In Section 8 we present a Vivid solution to a well-known puzzle in heterogeneous problem solving, while Section 9 illustrates the use of Vivid for reasoning about constraint problems (in this particular example, map coloring). Finally, in Section 10 we discuss related work, and Section 11 concludes. A Vivid system allowing for the formulation and solution of arbitrary seating puzzles of the sort described in Section 8 has been implemented, and can be downloaded along with a number of machine-readable examples of heterogeneous proofs in that domain, including the solution to the puzzle of Barwise and Etchemendy.[6]

2. Some design desiderata

We begin by listing, briefly, some conditions that Vivid was designed to meet. We then proceed to discuss each in more detail.

- *Combine symbolic and diagrammatic reasoning*. As is well-known in AI, some systems are geared toward reasoning that is exclusively symbolic in nature (e.g., resolution-based theorem proving). The visual counterparts to such systems are those that only allow diagrammatic reasoning, and are therefore devoid of formulas of the sort seen in propositional and predicate logic (e.g., traditional Venn diagrams). Vivid is intended to strike a marriage between these two extremes, by enabling *heterogeneous* reasoning: reasoning that has the tight step-by-step structure of formal, symbolic deduction, and yet at the same time makes productive use of diagrams.
- *Achieve diagrammatic generality*. Most diagrammatic reasoning systems are committed to what might be called *particularism*; i.e., they are suited to representing, and enabling reasoning over, a relatively small, fixed space of objects. One example is Peirce's α system, which provides a diagrammatic rendition of the propositional calculus. As such, this system does not provide the resources for analogically representing, say, circuits, seating arrangements around a dinner table, or arbitrary blocks-world configurations. To measure up to our goals for it, Vivid must reach an appreciable level of generality: It must be a framework for representing and reasoning heterogeneously over objects in arbitrary domains.
- *Enable mechanization*. Any particular instance of Vivid should be mechanizable. The diagrams should be readable and editable by machine, and the reasoning carried out with them should be certifiable by machine, and, at least in principle, automatically obtainable by machine as well. It is not enough to empower reasoners to refine their intuitions to some degree, or to merely externalize those intuitions in diagrams.
- *Develop meta-theory*. It is a customary expectation by now that the presentation of a logical system should be accompanied by—or at least be formulated with a rigor sufficient to allow the in-principle articulation of—meta-mathematical results. Such results often include soundness, completeness, decidability, and so on.

2.1. Heterogeneous representation and inference

Vivid, as indicated, is motivated by the desire to rigorously marry symbolic and diagrammatic reasoning. It is not difficult to motivate such a marriage. To begin, almost all diagrams contain textual labels and various others sentential elements in them. Such elements are crucial—if you remove all names and numbers from a city map, the map will be rendered useless for most practical purposes.

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

[5] We have omitted most proofs for space reasons; they appear in a longer technical report, at kryten.mm.rpi.edu/vivid/vivid.pdf.

[6] The URL is kryten.mm.rpi.edu/vivid/implementation/.

Sentential descriptions are particularly superior—and indeed often necessary—when it comes to expressing complex propositions. It is easy enough to depict an atomic piece of information such as is conveyed by the sentence "*a* is square" diagrammatically: We simply draw a square and label it *a*. But the proposition expressed by the statement "*a* is *not* square" is more challenging. How do we draw something that is not square? Certainly drawing it as a triangle will not do, nor as a cylinder or as any other particular shape. We need to stipulate a specific graphical convention for signifying that an object is not square. Perhaps we could draw a square with a line over it, to indicate negation, but if there are other attributes in addition to shape, say color, then would a line over a red square negate only squarehood or redness as well? What if we only wanted to say that it is not red? Apparently, any conventions we make will be ad hoc solutions[7] and can easily get out of hand. And while clever abstraction conventions can be introduced to express disjunctive information diagrammatically, sometimes with great visual clarity,[8] for most domains it will be very difficult to have enough abstraction conventions to be able to express *arbitrary* disjunctions. Existential and universal quantifications, being compactly expressed disjunctions and conjunctions, are even more powerful: "There is a knight on the chessboard" represents a huge number of possible chessboard configurations and excludes a huge number of others with just a few bits. In general, the issue is that in most interesting domains there are too many logical possibilities (models), while, due to physical spatial constraints, there is a much smaller number of possible diagrams. This discrepancy results in a certain tension. On the one hand, the discrepancy works to the cognitive advantage of diagrams, since the fewer the graphical possibilities, the clearer the diagrams. (Indeed, if we keep adding abstraction conventions in order to achieve a bijection between the class of diagrams and the class of set-theoretic models, we will probably end up ruining whatever analogical benefit we might have had originally. That is the case for Peirce's visual system for propositional logic, for instance, whose diagrams stand in a bijective relationship with logical sentences.[9]) On the other hand, the discrepancy can result in serious expressive limitations for diagrams.

Expressive limitations can sometimes lead to incorrect conclusions, since different models might be wrongly conflated (represented by the same diagram). This is a known issue, for example, with some Euler circles [20], as a consequence of Helly's theorem in convex topology [17]. A simple illustration of the problem, due to Lemon and Pratt [36],[10] is the following: Consider four sets A, B, C, and D, any three of which have a non-empty intersection:

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

These are four perfectly consistent premises. But any Euler diagram that tried to depict these premises would lead to the incorrect conclusion that all four sets have a non-empty intersection (i.e. that $A \cap B \cap C \cap D \neq \emptyset$), which does *not* follow from the premises.[11] The reason is that there is no way to draw four convex regions on the plane so that any three of the regions intersect without having all four of them intersect. Again, this is a consequence of Helly's theorem. Similar negative results hold for other diagrammatic ways of depicting sets and relationships between them, such as Englebretsen's [18] linear diagrams; see Lemon [35] for a thorough discussion.

The complexity of diagrammatic reasoning is another potential concern. Roughly, there are two types of diagrammatic inference. In one of them, exemplified by Euler circles, Venn diagrams, etc., inference is carried out by drawing appropriate diagrams. We then simply read off the appropriate bits of information from the constructed picture. This type of diagrammatic inference is summarized (rather crudely) by the slogan "If you can draw it, it holds."[12] In the second type of diagrammatic reasoning, inference is carried out in a more traditional sense, by deriving new diagrams from other diagrams that are given as premises, perhaps in tandem with given symbolic information (as in Vivid), or by extracting symbolic information from given diagrams. Computational complexity issues have been investigated more extensively for the former type of diagrammatic reasoning. For example, it has been realized that results obtained in studying the complexity of topological inference [23] have a direct bearing on the complexity of drawing diagrams such as Euler circles. It has been shown, e.g., that propositional reasoning with Euler sets is NP-hard, even though reasoning about the same domain can be done polynomially using other representations [35].

In our own work, diagrammatic inference is based on reasoning with incomplete information and has a strong model-theoretic flavor, proceeding essentially by model elimination. This can often make proof checking considerably more computationally intensive than it is in the purely sentential case, where proofs can be checked very efficiently. It would appear,

[7] Stenning [56] calls such conventions "abstraction tricks." We will be referring to them as "abstraction conventions." These are conventions intended to convey a complex meaning by arranging various visual tokens in certain fixed patterns. An example is the use of "X" marks and linking in Venn diagrams.

[8] An example is the judicious use of question-marks inside diagrams, an abstraction convention that is often used in Vivid diagrams.

[9] Shin and Lemon [53] make a similar point about Peirce's modification of Euler diagrams (the addition of X-sequences, etc.), writing that "the arbitrariness of its conventions and more confusing representations sacrificed the visual clarity which Euler's original system enjoys."

[10] Certain types of Euler circles are less restrictive than Lemon and Pratt's account. E.g., see Fish and Stapleton [22].

[11] As a countermodel, take $A = \{1, 2, 4\}$, $B = \{2, 3, 4\}$, $C = \{1, 3, 4\}$, and $D = \{1, 2, 3\}$.

[12] For instance, to check the validity of a syllogism with a Venn diagram, all we have to do is draw a figure that represents the premises of the syllogism. When finished, the picture itself will tell us whether or not the conclusion follows; nothing further needs to be done. Hence, inference in such cases stops with the representation of the premises. In customary reasoning, by contrast, inference only begins *after* the premises have been represented.

therefore, that visual inference, at least in some cases, can be significantly more expensive than corresponding sentential reasoning.[13]

For these and other reasons, a number of researchers have concluded, as have we, that logical reasoning frameworks must be *heterogeneous* or *hybrid* [7,41]; 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 [7] correctly observe, it would be impossible to construct a domain-independent framework for diagrammatic reasoning that relied on a specific type of diagrams. What makes a class of diagrams appropriate—i.e., good analogical representations—for certain problems might make them inappropriate for others. For example, at different times electrical engineers use state diagrams, circuit diagrams, and timing diagrams to represent and reason about hardware as needed by the appropriate viewpoint at hand (control, logic gates, or timing, respectively). There is no single type of diagram that is uniformly appropriate. This naturally leads us to the next point.

2.2. Diagrammatic generality

In an influential survey of formal methods, Rushby [48, p. 66] wrote the following about diagrams:

> The problem [with expressing formal specifications in diagrammatic or tabular notations] is that while diagrams and tables may be convenient ways to explain certain types of specification, they do not lend themselves to particularly effective mechanized reasoning and ... mechanically-supported analysis. Furthermore, different problem domains lend themselves to different types of diagrams or tables .... It follows that a formal specification method built around a particular diagrammatic or tabular notation may have rather restricted application, and limited mechanized support for general forms of analysis.

That is a valid and pressing concern. After all, formal symbolic logic would not be the success that it is if we had to invent a completely new logical theory (and meta-theory) and build a new implementation for every different application. Thankfully, the abstract syntax, semantics, and proof theory of first-order logic are the same regardless of whether we are talking about circuits, software, blocks worlds, or people.

Contrary to the widespread perception expressed in the foregoing quote, we believe that a similar level of generality can be attained for heterogeneous reasoning. There is no a priori reason why diagrammatic representation and inference have to be unduly specialized and ad hoc, and much of our work on Vivid has been intended to overcome the narrow specificity identified by Rushby. It turns out that much of what we do when we reason with a combination of diagrammatic and symbolic information does not depend on how the diagrams are drawn or even on what they mean in the context of a specific application domain. In this paper we identify what is common in a great variety of instances of heterogeneous reasoning, and proceed to factor it out and extrapolate it into general formal constructs. In the resulting framework, the type of diagrams used may vary from application to application, but the principles by which we reason with diagrammatic and symbolic information remain the same.

This is not unlike the separations that are familiar from traditional symbolic logic, where the vocabulary varies from application to application (e.g., we have different constant, relation, and function symbols as dictated by the problem domain), and the interpretation of the atomic formulas that we can build from that vocabulary will also vary, but the general principles by which we reason with such formulas do not change. In fact, 'first-order logic' denotes a framework allowing for an infinite number of instantiations (first-order languages with different symbol sets), and proof theories for first-order logic are pitched in a way that is agnostic as to what particular symbols are legislated. This situation makes it possible for different people to use different particular symbols, and yet do so with the knowledge that parsing technology can be easily created to distill such symbols to a fixed, underlying representation scheme that can be mechanically reasoned over. Such representation schemes are typically specified by use of meta-variables, which themselves carry no particular symbolic content. Nonetheless, it should not be said about such schemes that they are not really symbolic because they are bereft of particular symbols.

The situation with respect to Vivid is a close parallel. Vivid has a highly general and modular structure: A specific instance of it is obtained by fixing a class of diagrams (this is done by providing a diagram parser), and an interpretation of the diagrams through an appropriate attribute structure. This is akin to the way in which the CLP(X) scheme instantiates specific constraint logic programming languages by substituting different domains for X [29].

2.3. Mechanization

The capacity to follow and to evaluate a logical train of thought is considered to be one of the hallmarks of intelligence, so we want to replicate that capacity in machines. Of course we want machines to do more—we want them to creatively

[13] Of course in some contexts this can be viewed as an advantage, as time is usually traded for space. Vivid deductions, for instance, are typically much more compact than corresponding sentential deductions.

discover rational trains of thought. But, at the very least, we expect them to be able to inspect and assess the correctness of a given piece of reasoning. In the symbolic realm, this is standard fare. At least in principle, sentential proofs can now be mechanically evaluated in a fairly routine manner. If heterogeneous reasoning is to gain wide currency, we believe it imperative that heterogeneous proofs can likewise be mechanically certified by standard and well-understood techniques.[14]

2.4. Meta-theory

Little needs to be said about this requirement, according to which any logical system, and any broader framework of logical systems, must be characterized precisely enough to determine, at least eventually, whether or not certain key meta-properties are possessed by the systems in question. This requirement is particularly important when logical systems are intended for use in AI. For example, it is certainly of some practical import that we know first-order logic to be sound, complete, but only semi-decidable. Of course, for a given logical system, or class of such, the development of a mature meta-theory can take some time—but the system or class, from its inception, must allow for this development to unfold. For example, Frege's contribution consisted in no small part in the fact that his seminal specification of first-order logic allowed Henkin, decades later, to provide the completeness theorem. Vivid is proved sound and decidable in this paper, and is poised for the development of additional meta-theoretical results.

A final point to conclude this section: While Vivid has been designed to ensure satisfaction of these desiderata, other hybrid diagrammatic-symbolic systems, including in some cases ones quite different from Vivid, could certainly satisfy them as well. In the purely symbolic case, after all, there are infinitely many logics, and even if we restrict the class of symbolic logics to those asserted by various AI experts to have certain desirable attributes, the resulting list would be exceedingly large, and healthy discussion, well short of consensus, will persist.[15] We anticipate the parallel: A growing number of heterogeneous logics will arrive on the AI scene, each, we trust, contextualized by the set of desiderata it satisfies; and in some cases, inevitably, that set will depart somewhat from our own.

3. Notation

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

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

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

$$Pos(x, L) = \left\{ i \in \{1, \ldots, n\} \;\middle|\; x = x_i \right\}.$$

Thus, if x does not occur in L then $Pos(x, L) = \emptyset$. 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 $[y_1 y_2 \cdots y_m]$ 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 $[y_2 \cdots y_m]$. Strings (words) over an alphabet Σ are simply lists of elements of Σ. To adhere to customary string notation, we will usually denote the empty string by ϵ (instead of $[\,]$), and we will omit the enclosing brackets when writing non-empty strings, e.g. writing $a\,b\,c$ instead of $[abc]$.

For any set A, we write $\mathcal{P}_{fin}(A)$ 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.[16] 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 $A' \subseteq A$, $f \restriction A'$ denotes the restriction of f on A', i.e.,

$$f \restriction A' = \left\{ (x; y) \;\middle|\; f(x) = y \text{ and } x \in A' \right\}.$$

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

[14] Of course this is not to say that all diagram-infused problem solving does (or should) involve proofs, let alone proofs that should be encoded and checked in Vivid.

[15] E.g., some feel that in light of Lindström's Theorem, logic-based AI should not move beyond first-order logic; some feel that even first-order logic, given its undecidability, is too expressive, and that description logics (or even just simple subsets of the propositional calculus) are best; some feel that moving at least to quantified modal logics is necessary; and so on.

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

4. Attribute structures and systems

Very broadly, a diagram depicts a finite number of objects, and conveys (though perhaps only partially) the values of certain *observable attributes* of these objects. Of course what counts as an object and what counts as an attribute depends on our underlying theory and on our purposes, i.e., on *how* we choose to look at a diagram. That is, how we parse the raw pictorial information of a diagram varies widely, depending on the application at hand. But, assuming that we have fixed what constitutes an object and which observable object attributes we are interested in, the upshot of diagram parsing can be thought of as a mapping that assigns one or more values to every observable attribute of every object represented in the diagram. If exactly one value is assigned to every attribute and object, then the diagram is completely determined; there is no ambiguity or imprecision about it. Incomplete diagrammatic information is captured by assigning multiple values to a single object and attribute; e.g., if a diagram does not completely determine the color of an object o, we might assign a *set* of several color values to o, say, green, red, and blue. What follows is a formal development of these intuitions.

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

An attribute structure is thus a type of regular heterogeneous algebraic structure [38], without any operators, whose carriers are called "attributes." We will assume that $\mathcal{R}$ includes the identity relation on each attribute A_i: $\{(a; a) \mid a \in A_i\}$.

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

Definition 2. Let $\mathcal{A} = (\{A_1, \ldots, A_k\}; \mathcal{R})$ be an attribute structure. An **attribute system** based on $\mathcal{A}$, or $\mathcal{A}$-system for short, is a pair

$$\mathcal{S} = \big(\{s_1, \ldots, s_n\}; \mathcal{A}\big)$$

consisting of a finite number $n > 0$ of **objects** $s_1, \ldots, s_n$ and $\mathcal{A}$. An attribute of $\mathcal{A}$ may include some (perhaps all) of the objects $s_1, \ldots, s_n$. If at least one attribute in $\mathcal{A}$ includes some of these objects,[17] $\mathcal{S}$ is called **automorphic**. We refer to the product $n \cdot |\mathcal{A}|$ as the system's **power**.

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

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

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

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:

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

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:

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

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

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

and *floor* is distinct from A, B, and C. This system is also automorphic.

[17] That is, if $A_i \cap \{s_1, \ldots, s_n\} \neq \emptyset$ for some $i \in \{1, \ldots, k\}$.

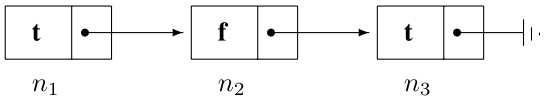


Fig. 1. A linked list world.

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.,

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

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.

A system for an attribute structure with a finite basis has

$$\prod_{i=1}^{k} \big[2^{|A_i|} - 1\big]^n$$

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

Example 4. Consider the single-clock system of Example 1:

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

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

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

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:

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

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

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

This state can also be expressed by writing

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

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

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

Example 5. Consider the linked-list system of Example 2. The state

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

depicts the world shown in Fig. 1. The state

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

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.

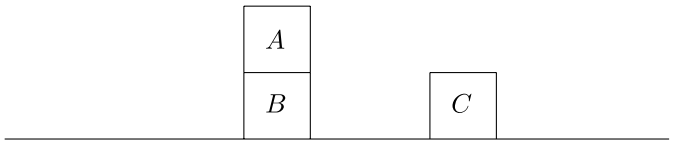


Fig. 2. A blocks world.

Example 6. Consider the blocks world system of Example 3. The state

$$pos(A) = B, \quad pos(B) = floor, \quad pos(C) = floor \tag{1}$$

depicts the blocks world shown in Fig. 2. The state

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

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

Here we assume that we are only interested in how the blocks are stacked, not in the left-of relation. So, in the case of Fig. 2, if C were on the floor to the left of A and B instead of the right, we would still get the same state, (1). Of course we could enrich our attribute structure to take such additional information into account, if doing so was deemed important. In general, most abstract representations of a diagram will discard certain bits of information conveyed by the diagram as irrelevant. Which aspects of a diagram are considered essential and which are not depends on the domain at hand and on our purposes. In Venn diagrams, for instance, the sizes of the regions is immaterial. Likewise, if we have two intersecting circles A and B, it is immaterial whether A is to the left of B or vice versa. The issue, of course, is not with the representations but with the diagrams themselves. As discussed in the introduction, diagrams tend to be overly specific and thus we usually need to ignore at least some of their aspects.[18]

We might (rather loosely) think of system states as mental models of situations [31], representing various states of knowledge ranging from completely specific to completely indeterminate.

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

$$w(l_1, s_1)w(l_2, s_1) \cdots w(l_k, s_1) \cdots w(l_1, s_n) \cdots w(l_k, s_n).$$

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

Example 7. Assuming that $data \prec_l next$ and that $n_1 \prec_s n_2 \prec_s n_3$, the world described in Example 5 (and shown in Fig. 1) can be viewed as the six-element string $\mathbf{t}\, n_2\, \mathbf{f}\, n_3\, \mathbf{t}\, null$.

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 $j = 1, \ldots, n$.[19] 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:

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

The state

$$\begin{aligned} &hours'(c_1) = \{13, 14\}, \\ &minutes'(c_1) = \{55\}, \\ &hours'(c_2) = \{6, 7\}, \\ &minutes'(c_2) = \{9, 10\}, \end{aligned} \tag{2}$$

[18] For instance, in the usual diagrammatic proof of the Pythagorean theorem, we *must* ignore the relative lengths of the triangle's sides in order to ensure that we are reasoning about an "arbitrary" triangle, even though any picture of a triangle will necessarily depict specific lengths for its sides.

[19] The terminology sounds somewhat counterintuitive, since an extension of a state is one that assigns *fewer* values to each object/attribute pair, thereby increasing the overall information content. This is similar to the terminology of object-oriented class hierarchies, where we might say that the concept *human* is an extension of *mammal* to mean that the former is in fact a subset of the latter.

is an extension (a proper one) of the state

$$\begin{aligned} &\mathit{hours}(c_1) = \{13, 14, 15\},\\ &\mathit{minutes}(c_1) = \{55\},\\ &\mathit{hours}(c_2) = \{6, 7\},\\ &\mathit{minutes}(c_2) = \{9, 10, 11\}. \end{aligned} \tag{3}$$

Recalling that a world can be encoded as a string, we may view a system state σ as a language, namely as the set of all strings w such that $w \sqsubseteq \sigma$. For instance, the second state described in Example 5 can be understood as the following set of eight strings:

$$\{\mathbf{t}n_2\,\mathbf{t}n_1\,\mathbf{f}\mathit{null}, \mathbf{t}n_2\,\mathbf{t}n_3\,\mathbf{f}\mathit{null}, \mathbf{t}n_2\,\mathbf{f}n_1\,\mathbf{f}\mathit{null}, \mathbf{t}n_2\,\mathbf{f}n_3\,\mathbf{f}\mathit{null}, \mathbf{f}n_2\,\mathbf{t}n_1\,\mathbf{f}\mathit{null}, \mathbf{f}n_2\,\mathbf{t}n_3\,\mathbf{f}\mathit{null}, \mathbf{f}n_2\,\mathbf{f}n_1\,\mathbf{f}\mathit{null}, \mathbf{f}n_2\,\mathbf{f}n_3\,\mathbf{f}\mathit{null}\}.$$

Note that all such languages are finite, and therefore regular. This is important in representing system states as minimal acyclic deterministic finite automata (ADFA), which support highly efficient insertions and membership tests.

Definition 5. Two system states σ_1 and σ_2 are **disjoint** iff there is no world w such that $w \sqsubseteq \sigma_1$ and $w \sqsubseteq \sigma_2$.

Viewing σ_1 and σ_2 as languages, the two are disjoint iff they are set-theoretically disjoint, i.e., iff $\sigma_1 \cap \sigma_2 = \emptyset$. Any two distinct worlds are automatically disjoint.[20] Both perspectives on system states (as indexed sets of ascriptions and as languages) are useful, although the second one is not widely used in this paper. For instance, we might speak of a world w as *extending* a certain state σ, $w \sqsubseteq \sigma$, viewing w and σ as sets of ascriptions; or we might speak of w as *belonging* to σ, $w \in \sigma$, viewing the latter as a language and w as a string in that language.

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

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

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

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

This is precisely the least upper bound of the two states w.r.t. the ordering $\sqsubseteq$. But a meet operator $\sqcap$ would indicate conjunction, and conjoining diagrams with contradictory information is not pictorially meaningful. For instance, if an object s has a round shape in diagram σ_1 and a square shape in σ_2:

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

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

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

then we would have $\sigma_1 \sqcap \sigma_2(\mathit{shape}, s) = \emptyset$, indicating that s has no shape (!) in $\sigma_1 \sqcap \sigma_2$. Sententially, we can easily write a formula that asserts

$$\mathit{Round}(s) \wedge \mathit{Square}(s),$$

but, diagrammatically, we cannot *draw* a square circle (but see the next paragraph). Likewise, we can very well write a sentence that asserts that it is currently 5:15 a.m. and also that it is currently 11:30 p.m., but if we look at the face of a clock we will not see it displaying both times. There are no inconsistent diagrams, meaning that there is no diagram that can both ascribe and not ascribe a certain observable attribute to one of its elements,[21] and this is, in turn, related 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.

Having said this, we certainly concede that it is easy enough to diagrammatically denote (or at least suggest) inconsistency and negation, and, given this, to conjoin a diagram with that which so denotes. After all, in his α existential graph

[20] When a world is viewed as a language, it is obviously understood as a singleton.

[21] Optical illusions should not be confused with logical inconsistency.

system, Peirce uses Ⓟ to denote the negation of p, and following his lead, we could decide to use, say, ⦿ to indicate inconsistency. But such techniques forfeit what, following others, we have adopted (Section 1) as the distinguishing mark of good diagrams: that they refer analogically (or homomorphically). At least by our lights, it is difficult to see how ⦿, or anything of the sort, *structurally corresponds* to inconsistency. Likewise, it might be said that a Venn diagram with a region at once both shaded and "Xed" denotes inconsistency. But this move does not produce something that *resembles* inconsistency; it merely produces something that can be agreed by convention to suggest inconsistency. Note further that such a diagram is by itself perfectly consistent and hence can be represented in Vivid quite straightforwardly: It is a collection of minimal plane regions [24, p. 33], each of which has a number of observable attributes, e.g., a token such as $\times$ or $\circ$ (or none, indicating lack of information about the cardinality of the region). It is not the diagram itself that is inconsistent—again, that would require the simultaneous presence and absence of an observable property, which is physically impossible. Rather, it is our interpretation conventions that make us regard the diagram as depicting an inconsistency, albeit in a non-analogical manner.

In principle, by descending to a sufficiently low level of detail, any diagram whatsoever can be represented in Vivid as a system state, simply by treating the diagram as a raster image—an arbitrarily large grid of raw pixels. The objects of the system would be the pixel locations, there would be only one attribute, intensity, and a system world would map each object (pixel location on the grid) to a particular intensity. Of course, reasoning with such a low-level diagrammatic representation would be severely impractical because, by and large, pixels are not analogical representations (e.g., relations between pixels are generally not reflective of relations between the objects of interest).

5. Interpreting relational languages over 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 interpretation** 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 the following 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:

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

   (where we might have $m \neq n$); and
2. a list of m pairs

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

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

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

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

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

and an attribute interpretation 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 computable partial function ρ from C to $\{s_1, \ldots, s_n\}$; while a **variable assignment** is a computable 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 require $Dom(\rho)$ to be computable, for any ρ. Further, when ρ is finite, $Dom(\rho)$ should be effectively obtainable from ρ. We 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.[22] We write $FV(F)$ for the set of variables that occur free in a formula F, and $CS(F)$ for the set of all constant symbols that occur in 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. We write $t^{\rho,\chi}\uparrow$ to indicate that $t^{\rho,\chi}$ is undefined, and $t^{\rho,\chi}\downarrow$ to indicate that it is defined.

[22] Consult any textbook on mathematical logic for details.

By a **named state** we 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 $\sigma' \sqsubset \sigma$ 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 $(\sigma;\rho)$ is called a **world** iff σ is a world (every ascription of σ is a valuation) and ρ is total.[23] 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 mean a finite set of formulas. We write $FV(\beta)$ and $CS(\beta)$ to denote the set of all variables that have free occurrences in some element of β, and the set of all constant symbols that appear in some element of β, respectively. 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 $(\sigma_1;\rho_1) \sqsubseteq (\sigma_2;\rho_2)$ is decidable whenever at least one ρ_i is finite.

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 an arbitrary 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 defining 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.

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

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

$$\mathcal{V}^I_{(w;\rho)/\chi}[\textbf{true}] = \textbf{true} \quad \text{and} \quad \mathcal{V}^I_{(w;\rho)/\chi}[\textbf{false}] = \textbf{false}. \tag{4}$$

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

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

We have:

$$\mathcal{V}^I_{(w;\rho)/\chi}\big[R^I(t_1,\ldots,t_n)\big] = \begin{cases} \textbf{unknown} & \text{if } \exists\, k \in \{1,\ldots,m\}\,.\, t_{j_k}^{\rho,\chi} \uparrow; \\ \textbf{true} & \text{if } R^I(w(l_{i_1}, t_{j_1}^{\rho,\chi}),\ldots,w(l_{i_m}, t_{j_m}^{\rho,\chi})); \\ \textbf{false} & \text{if } \neg R^I(w(l_{i_1}, t_{j_1}^{\rho,\chi}),\ldots,w(l_{i_m}, t_{j_m}^{\rho,\chi})). \end{cases} \tag{5}$$

The semantics of the remaining connectives are given in accordance with the strong 3-valued Kleene scheme, as shown in Fig. 3. Conditionals $F \Rightarrow G$ are treated as disjunctions $\neg F \vee G$, and biconditionals $F \Leftrightarrow G$ as conjunctions $(F \Rightarrow G) \wedge (G \Rightarrow F)$.

Note that occurrences of symbols such as $\forall$ and $\exists$ on the right-hand sides of (5)–(10) occur as part of our metalanguage and should not be confused with object-level occurrences of these symbols in Vivid formulas. We will continue to use object-level symbols in different capacities without explicitly calling attention to the distinction; the context will always clarify the use.

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

$$I_{(\sigma;\rho)/\chi}(F) = \begin{cases} \textbf{true} & \text{if } \mathcal{V}^I_{(w;\rho)/\chi}[F] = \textbf{true} \text{ for every } w \sqsubseteq \sigma; \\ \textbf{false} & \text{if } \mathcal{V}^I_{(w;\rho)/\chi}[F] = \textbf{false} \text{ for every } w \sqsubseteq \sigma; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{11}$$

Example 8. Consider the signature $\Sigma_1 = (\mathsf{C}_{clock};\mathsf{R}_{clock};\mathsf{V}_{clock})$ where the set of constant symbols is

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

the set of variables is $\mathsf{V}_{clock} = \{x, y, z, x_1, y_1, z_1, \ldots\}$, and the set of relation symbols is

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

[23] 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.

$$\mathcal{V}^I_{(w;\,\rho)/\chi}[\neg F] = \begin{cases} \textbf{true} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi}[F] = \textbf{false}; \\ \textbf{false} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi}[F] = \textbf{true}; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{6}$$

$$\mathcal{V}^I_{(w;\,\rho)/\chi}[F_1 \wedge F_2] = \begin{cases} \textbf{true} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi}[F_1] = \textbf{true} \text{ and } \mathcal{V}^I_{(w;\,\rho)/\chi}[F_2] = \textbf{true}; \\ \textbf{false} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi}[F_1] = \textbf{false} \text{ or } \mathcal{V}^I_{(w;\,\rho)/\chi}[F_2] = \textbf{false}; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{7}$$

$$\mathcal{V}^I_{(w;\,\rho)/\chi}[F_1 \vee F_2] = \begin{cases} \textbf{true} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi}[F_1] = \textbf{true} \text{ or } \mathcal{V}^I_{(w;\,\rho)/\chi}[F_2] = \textbf{true}; \\ \textbf{false} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi}[F_1] = \textbf{false} \text{ and } \mathcal{V}^I_{(w;\,\rho)/\chi}[F_2] = \textbf{false}; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{8}$$

$$\mathcal{V}^I_{(w;\,\rho)/\chi}[\forall\, v\,.\, F] = \begin{cases} \textbf{true} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi[v\mapsto s_i]}[F] = \textbf{true} \text{ for all } i \in \{1,\ldots,n\}; \\ \textbf{false} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi[v\mapsto s_i]}[F] = \textbf{false} \text{ for some } i \in \{1,\ldots,n\}; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{9}$$

$$\mathcal{V}^I_{(w;\,\rho)/\chi}[\exists\, v\,.\, F] = \begin{cases} \textbf{true} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi[v\mapsto s_i]}[F] = \textbf{true} \text{ for some } i \in \{1,\ldots,n\}; \\ \textbf{false} & \text{if } \mathcal{V}^I_{(w;\,\rho)/\chi[v\mapsto s_i]}[F] = \textbf{false} \text{ for all } i \in \{1,\ldots,n\}; \\ \textbf{unknown} & \text{otherwise.} \end{cases} \tag{10}$$

Fig. 3. Strong Kleene semantics of complex formulas with respect to individual worlds.

with PM, AM unary and Ahead, Behind binary.

Consider now the attribute structure

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

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: $R_1(h) \Leftrightarrow h > 11$, $R_2(h) \Leftrightarrow h \leqslant 11$,

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

and

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

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 $\texttt{PM}^I = R_1$, $\texttt{AM}^I = R_2$, $\texttt{Ahead}^I = R_3$, $\texttt{Behind}^I = R_4$, and:

$$\mathit{Prof}(\texttt{PM}) = \big[(\mathit{hours}, 1)\big];$$
$$\mathit{Prof}(\texttt{AM}) = \big[(\mathit{hours}, 1)\big];$$
$$\mathit{Prof}(\texttt{Ahead}) = \big[(\mathit{hours}, 1), (\mathit{minutes}, 1), (\mathit{hours}, 2), (\mathit{minutes}, 2)\big];$$
$$\mathit{Prof}(\texttt{Behind}) = \big[(\mathit{hours}, 1), (\mathit{minutes}, 1), (\mathit{hours}, 2), (\mathit{minutes}, 2)\big].$$

Example 9. Consider the system $(\{s_1, s_2\}; \mathit{Clock})$, where *Clock* is the attribute structure of Example 8. Let σ be the following state of this system:

$$\mathit{hours}(s_1) = \{9, 13\},$$
$$\mathit{minutes}(s_1) = 12,$$
$$\mathit{hours}(s_2) = 8,$$
$$\mathit{minutes}(s_2) = 27,$$

and let ρ be the partial constant assignment that maps $\mathtt{c}_1$ to s_1 and $\mathtt{c}_2$ to s_2. There are exactly two worlds w_1 and w_2 in σ, where $w_1(\mathit{hours}, s_1) = 9$ and $w_2(\mathit{hours}, s_1) = 13$, with w_1 and w_2 in agreement everywhere else. 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 (11) tells us that in order to have

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

we must have

$$\mathcal{V}^I_{(w;\rho)/\chi}\big[\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)\big] = \textbf{true}$$

for every $w \sqsubseteq \sigma$. It is straightforward to verify that this is the case: We have

$$\mathcal{V}^I_{(w_1;\rho)/\chi}\big[\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)\big] = \textbf{true}$$

because $R_3(9, 12, 8, 27)$; and we also have

$$\mathcal{V}^I_{(w_2;\rho)/\chi}\big[\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)\big] = \textbf{true},$$

given that $R_3(13, 12, 8, 27)$.

Lemma 2. *If $\mathit{Dom}(\rho) \supseteq \mathit{CS}(F)$ then $\mathcal{V}^I_{(w;\rho)/\chi}[F] \neq \textbf{unknown}$.*

Proof. A straightforward induction on the structure of F. □

In practice, $\mathcal{V}^I_{(w;\rho)/\chi}[F]$ is almost always **true** or **false**. By Lemma 2, there is only one way in which it can be **unknown**, namely, if F contains constant symbols which are not in the domain of ρ.[24] Even then, the result might not be **unknown**. Consider, for instance, a formula such as $R(c_1, c_2)$. If, as the first clause of (5) indicates, the profile of R involves only the first argument position, then $\mathcal{V}^I_{(w;\rho)/\chi}[R(c_1, c_2)]$ might return **true** or **false** even when $c_2 \notin \mathit{Dom}(\rho)$. Alternatively, a conjunction $F_1 \wedge F_2$ (or disjunction $F_1 \vee F_2$) might return **false** (respectively, **true**) even if F_2 contains undefined constant symbols, etc. But certainly as long as every constant symbol that appears in the formula F is covered by ρ, the value $\mathcal{V}^I_{(w;\rho)/\chi}[F]$ will not be **unknown**.

Lemma 3. (a) *If $\mathcal{V}^I_{(w;\rho)/\chi}[F] \neq \textbf{unknown}$ and $\rho \subseteq \rho'$, then $\mathcal{V}^I_{(w;\rho')/\chi}[F] = \mathcal{V}^I_{(w;\rho)/\chi}[F]$.* (b) *If $\mathit{Dom}(\rho) \supseteq \mathit{CS}(F)$ and $\rho \subseteq \rho'$ then $\mathcal{V}^I_{(w;\rho)/\chi}[F] = \mathcal{V}^I_{(w;\rho')/\chi}[F]$.*

Proof. By structural induction on F. ((b) also follows from (a) and Lemma 2.) □

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

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

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

The following result is readily proved by induction on the structure of F. It is the 3-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 5. *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 6. *Let R be a relation symbol of arity n and profile $[(l_{i_1}; j_1) \cdots (l_{i_m}; j_m)]$, and suppose that $t^{\rho,\chi}_{j_k}$ is defined for every $k = 1, \ldots, n$. Then*:

(a) $I_{(\sigma;\rho)/\chi}\big(R(t_1, \ldots, t_n)\big) = \textbf{true}$ *iff* $\forall a_1 \in \sigma\big(l_{i_1}, t^{\rho,\chi}_{j_1}\big) \cdots \forall a_m \in \sigma\big(l_{i_m}, t^{\rho,\chi}_{j_m}\big)\,.\, R^I(a_1, \ldots, a_m)$;

(b) $I_{(\sigma;\rho)/\chi}\big(R(t_1, \ldots, t_n)\big) = \textbf{false}$ *iff* $\forall a_1 \in \sigma\big(l_{i_1}, t^{\rho,\chi}_{j_1}\big) \cdots \forall a_m \in \sigma\big(l_{i_m}, t^{\rho,\chi}_{j_m}\big)\,.\, \neg R^I(a_1, \ldots, a_m)$;

(c) $I_{(\sigma;\rho)/\chi}\big(R(t_1, \ldots, t_n)\big) = \textbf{unknown}$ *iff* $\big[\exists\, a_1 \in \sigma\big(l_{i_1}, t^{\rho,\chi}_{j_1}\big) \cdots \exists\, a_m \in \sigma\big(l_{i_m}, t^{\rho,\chi}_{j_m}\big)\,.\, R^I(a_1, \ldots, a_m)$ *and* $\exists\, a_1 \in \sigma\big(l_{i_1}, t^{\rho,\chi}_{j_1}\big) \cdots \exists\, a_m \in \sigma\big(l_{i_m}, t^{\rho,\chi}_{j_m}\big)\,.\, \neg R^I(a_1, \ldots, a_m)\big]$.

[24] Since χ is always total, there is no issue of free variables of F not in the domain of χ.

The following is a direct consequence of the finite size of the ascription values, the finite number of system objects, and Lemma 5.

Theorem 7. *$I_{(\sigma;\rho)/\chi}$ is computable for any named state $(\sigma;\rho)$ and variable assignment χ.*

Lemma 8. (a) $I_{(\sigma;\rho)/\chi}(\neg F) = \textbf{true}$ *iff* $I_{(\sigma;\rho)/\chi}(F) = \textbf{false}$; *and* (b) $I_{(\sigma;\rho)/\chi}(\neg F) = \textbf{false}$ *iff* $I_{(\sigma;\rho)/\chi}(F) = \textbf{true}$. *Therefore,*

$$I_{(\sigma;\rho)/\chi}(\neg F) = \textbf{unknown} \quad \textit{iff} \quad I_{(\sigma;\rho)/\chi}(F) = \textbf{unknown}.$$

Lemma 9. (a) $I_{(\sigma;\rho)/\chi}(F_1 \wedge F_2) = \textbf{true}$ *iff* $I_{(\sigma;\rho)/\chi}(F_1) = \textbf{true}$ *and* $I_{(\sigma;\rho)/\chi}(F_2) = \textbf{true}$.

(b) *If* $I_{(\sigma;\rho)/\chi}(F_1) = \textbf{false}$ *or* $I_{(\sigma;\rho)/\chi}(F_2) = \textbf{false}$ *then* $I_{(\sigma;\rho)/\chi}(F_1 \wedge F_2) = \textbf{false}$. *However, the converse is not true. In particular, we may have*:

$$I_{(\sigma;\rho)/\chi}(F_1 \wedge F_2) = \textbf{false}, \quad I_{(\sigma;\rho)/\chi}(F_1) = \textbf{unknown}, \quad I_{(\sigma;\rho)/\chi}(F_2) = \textbf{unknown}.$$

(c) *If* $\{I_{(\sigma;\rho)/\chi}(F_1), I_{(\sigma;\rho)/\chi}(F_2)\} = \{\textbf{true}, \textbf{unknown}\}$ *then* $I_{(\sigma;\rho)/\chi}(F_1 \wedge F_2) = \textbf{unknown}$.
(d) *If* $I_{(\sigma;\rho)/\chi}(F_1) = \textbf{unknown}$ *and* $I_{(\sigma;\rho)/\chi}(F_2) = \textbf{unknown}$ *then* $I_{(\sigma;\rho)/\chi}(F_1 \wedge F_2) \neq \textbf{true}$.

The following result is the dual of Lemma 9.

Lemma 10. (a) $I_{(\sigma;\rho)/\chi}(F_1 \vee F_2) = \textbf{false}$ *iff* $I_{(\sigma;\rho)/\chi}(F_1) = \textbf{false}$ *and* $I_{(\sigma;\rho)/\chi}(F_2) = \textbf{false}$.

(b) *If* $I_{(\sigma;\rho)/\chi}(F_1) = \textbf{true}$ *or* $I_{(\sigma;\rho)/\chi}(F_2) = \textbf{true}$ *then* $I_{(\sigma;\rho)/\chi}(F_1 \vee F_2) = \textbf{true}$. *However, the converse is not true. In particular, we may have*:

$$I_{(\sigma;\rho)/\chi}(F_1 \vee F_2) = \textbf{true}, \quad I_{(\sigma;\rho)/\chi}(F_1) = \textbf{unknown}, \quad I_{(\sigma;\rho)/\chi}(F_2) = \textbf{unknown}.$$

(c) *If* $\{I_{(\sigma;\rho)/\chi}(F_1), I_{(\sigma;\rho)/\chi}(F_2)\} = \{\textbf{false}, \textbf{unknown}\}$ *then* $I_{(\sigma;\rho)/\chi}(F_1 \vee F_2) = \textbf{unknown}$.
(d) *If* $I_{(\sigma;\rho)/\chi}(F_1) = \textbf{unknown}$ *and* $I_{(\sigma;\rho)/\chi}(F_2) = \textbf{unknown}$ *then* $I_{(\sigma;\rho)/\chi}(F_1 \vee F_2) \neq \textbf{false}$.

Lemma 11. (a) $I_{(\sigma;\rho)/\chi}(\forall v \,.\, F) = \textbf{true}$ *iff* $I_{(\sigma;\rho)/\chi[v \mapsto s_i]}(F) = \textbf{true}$ *for every* $i = 1, \ldots, n$.

(b) *If* $I_{(\sigma;\rho)/\chi[v \mapsto s_i]}(F) = \textbf{false}$ *for some system object* s_i, *then* $I_{(\sigma;\rho)/\chi}(\forall v \,.\, F) = \textbf{false}$. *However, the converse does not hold. In particular, we may have* $I_{(\sigma;\rho)/\chi}(\forall v \,.\, F) = \textbf{false}$ *even though* $I_{(\sigma;\rho)/\chi[v \mapsto s_i]}(F) \neq \textbf{false}$ *for every* $i = 1, \ldots, n$.

The fact that universal and existential generalizations have the same logical structure as conjunctions and disjunctions, respectively, accounts for the similarity between Lemma 11 and Lemma 9, whose contents are essentially identical. The only ostensible difference between the two is part (c) of Lemma 9, which claims that the conjunction of **true** and **unknown** components results in **unknown**, and which has no direct analogue in Lemma 11.[25] Indeed, the counterexample in the proof of Lemma 11 depicts a situation where every instance of a universal generalization is either **true** or **unknown** and yet the generalization itself is **false**: $I_{(\sigma;\rho)/\chi}(\forall v \,.\, F) = \textbf{false}$ even though $I_{(\sigma;\rho)/\chi[v \mapsto s_i]}(F) \neq \textbf{false}$ for all i. The disparity here is only apparent. The issue is that conjunctions are always binary whereas universal quantifications may have an arbitrarily large number of instances. In fact there *is* an analogue of part (c) for universal generalizations, namely: If there are only two system objects s_1 and s_2, and $I_{(\sigma;\rho)/\chi[v \mapsto s_1]}(F) = \textbf{true}$ while $I_{(\sigma;\rho)/\chi[v \mapsto s_2]}(F) = \textbf{unknown}$, then $I_{(\sigma;\rho)/\chi}(\forall v \,.\, F) = \textbf{unknown}$. (The verification of this is an easy exercise.) But since this result has very limited applicability (only when there are two system objects), it is not worth stating as a lemma. Conversely, the same type of situation produced by the aforementioned counterexample would obtain for conjunctions if they were allowed to have arbitrarily many components, e.g., as in $F = F_1 \wedge F_2 \wedge F_3$. Then if $I_{(\sigma;\rho)/\chi}(F_1) = \textbf{true}$ but $I_{(\sigma;\rho)/\chi}(F_2) = I_{(\sigma;\rho)/\chi}(F_3) = \textbf{unknown}$, we could well have $I_{(\sigma;\rho)/\chi}(F) = \textbf{false}$, because the two unknowns (F_2 and F_3) might result in **false**, which will then of course falsify the entire conjunction. Loosely speaking, every time two or more unknowns are combined conjunctively, whether in a conjunction proper or in a universal generalization, the result might be **false**.

The following is the analogue of the preceding lemma for existential generalizations, corresponding to Lemma 10:

Lemma 12. (a) $I_{(\sigma;\rho)/\chi}(\exists v \,.\, F) = \textbf{false}$ *iff* $I_{(\sigma;\rho)/\chi[v \mapsto s_i]}(F) = \textbf{false}$ *for every* $i = 1, \ldots, n$.

(b) *If* $I_{(\sigma;\rho)/\chi[v \mapsto s_i]}(F) = \textbf{true}$ *for some* $i \in \{1, \ldots, n\}$, *then*

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

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

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

[25] Part (d) of Lemma 9 has a straightforward analogue in the case of Lemma 11, which was not stated explicitly in the latter because it is rather trivial.

even though

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

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

Definition 6. A world $(w;\widehat{\rho})$ **satisfies a formula** F w.r.t. a variable assignment χ iff

$$\mathcal{V}^I_{(w;\widehat{\rho})/\chi}[F] = \textbf{true}.$$

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

$$\big(\beta \cup \beta';(\sigma;\rho)\big) \models (\sigma';\rho').$$

Lemma 14. *If* $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$ *and* $(\beta;(\sigma';\rho')) \models F$ *then* $(\beta;(\sigma;\rho)) \models F$.

Lemma 15. $(\beta;(\sigma;\rho)) \models (\sigma;\rho)$.

Lemma 16. $(\beta \cup \{\textbf{false}\};(\sigma;\rho)) \models (\sigma';\rho')$.

Lemma 17. *If* $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$ *and* $(\sigma';\rho') \sqsubseteq (\sigma'';\rho'')$ *then* $(\beta;(\sigma;\rho)) \models (\sigma'';\rho'')$.

Corollary 18 *(Widening is sound). If* $(\sigma;\rho) \sqsubseteq (\sigma';\rho')$ *then* $(\beta;(\sigma;\rho)) \models (\sigma';\rho')$.

Definition 7. Suppose that $(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m) \overset{\infty}{\sqsubset} (\sigma;\rho)$, for finite $Dom(\rho)$, 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 $(\sigma;\rho) \Vdash_\beta \{(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m)\}$, iff

$$\big(\beta;(\sigma;\rho)\big) \models (\sigma_i;\rho_i) \quad \text{for some } i \in \{1,\ldots,m\};$$

i.e., iff for all worlds $(w;\widehat{\rho})$ and variable assignments χ, if

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

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

Theorem 19. *The entailment relation of Definition* 7 *is decidable. That is, there is an algorithm that will take any assumption base* β *and any* $m+1$ *named states* $(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m),(\sigma;\rho)$ *such that*

$$(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m) \overset{\infty}{\sqsubset} (\sigma;\rho),$$

for $m > 0$ *and finite* $Dom(\rho)$, *and will determine whether or not* $(\sigma;\rho) \Vdash_\beta \{(\sigma_1;\rho_1),\ldots,(\sigma_m;\rho_m)\}$.

Proof. Let F be the conjunction of all the formulas in β. Let $\rho'_1,\ldots,\rho'_d$ be all and only the constant assignments on $Dom(\rho_1) \cup \cdots \cup Dom(\rho_m) \cup CS(F)$ that are supersets of ρ. There are n^b such assignments, where

$$\begin{aligned} b &= \big|\big[Dom(\rho_1) \cup \cdots \cup Dom(\rho_m) \cup CS(F)\big] \setminus Dom(\rho)\big| \\ &= \big|\big[Dom(\rho_1) \setminus Dom(\rho)\big] \cup \cdots \cup \big[Dom(\rho_m) \setminus Dom(\rho)\big] \cup \big[CS(F) \setminus Dom(\rho)\big]\big|, \end{aligned}$$

and they can be mechanically enumerated, given that $CS(F) \setminus Dom(\rho)$ and $Dom(\rho_i) \setminus Dom(\rho)$ are finite, $i = 1,\ldots,m$. Moreover, let $\psi_1,\ldots,\psi_e$ be all distinct functions from $FV(F)$ to the set of all system objects, and let $\chi_1,\ldots,\chi_e$ be arbitrary variable assignments such that $\chi_i \restriction FV(F) = \psi_i$ for every $i = 1,\ldots,e$. Clearly, such variable assignments can also be mechanically generated. Now let C' be the following decidable condition:

$$\forall w \sqsubseteq \sigma\,.\,\forall i \in \{1,\ldots,d\}\,.\,\forall j \in \{1,\ldots,e\}\,.\,\textit{if } \mathcal{V}^I_{(w;\rho'_i)/\chi_j}[F] = \textbf{true} \quad \textit{then } \exists\, z \in \{1,\ldots,m\}\,.\,(w;\rho'_i) \models (\sigma_z;\rho_z),$$

and let C be the more general condition that appears in Definition 7:

$$\forall w\,.\,\forall \widehat{\rho}\,.\,\forall \chi\,.\,\textit{if} (w;\widehat{\rho}) \models_\chi \big(\beta;(\sigma;\rho)\big) \quad \textit{then } \exists\, z \in \{1,\ldots,m\}\,.\,(w;\widehat{\rho}) \models (\sigma_z;\rho_z),$$

and which quantifies over *all* worlds and constant and variable assignments.

In what follows we show that C' is both necessary and sufficient for C:

- *C implies C'*: Assume C. To show C', pick any $w \sqsubseteq \sigma$, ρ_i', and χ_j, and suppose that

$$\mathcal{V}^I_{(w;\rho_i')/\chi_j}[F] = \textbf{true}. \tag{12}$$

Let $\widehat{\rho}$ be any total constant assignment that extends ρ_i', so that

$$\widehat{\rho} \supseteq \rho_i'. \tag{13}$$

Since $\rho_i' \supseteq \rho$, we have

$$\widehat{\rho} \supseteq \rho. \tag{14}$$

Further, since $Dom(\rho_i') \supseteq CS(F)$, (13), (12), and Lemma 3 yield

$$\mathcal{V}^I_{(w;\widehat{\rho})/\chi_j}[F] = \textbf{true}, \tag{15}$$

which is to say

$$(w;\widehat{\rho}) \models_{\chi_j} \beta. \tag{16}$$

Moreover, since $w \sqsubseteq \sigma$ and $\widehat{\rho} \supseteq \rho$, we have

$$(w;\widehat{\rho}) \models (\sigma;\rho). \tag{17}$$

Hence, by (16) and (17), $(w;\widehat{\rho}) \models_{\chi_j} (\beta;(\sigma;\rho))$. Therefore, by C, we infer that $(w;\widehat{\rho}) \models (\sigma_z;\rho_z)$ for some $z \in \{1,\ldots,m\}$, i.e.,

$$w \sqsubseteq \sigma_z \tag{18}$$

and

$$\widehat{\rho} \supseteq \rho_z. \tag{19}$$

But $\widehat{\rho} \restriction Dom(\rho_z) = \rho_i' \restriction Dom(\rho_z)$, since $\widehat{\rho} \supseteq \rho_i'$ and $Dom(\rho_i') \supseteq Dom(\rho_z)$. Therefore, (19) gives $\rho_i' \supseteq \rho_z$, and hence, by (18), $(w;\rho_i') \models (\sigma_z;\rho_z)$.

- *C' implies C*: Assume C'. To show C, consider any $(w;\widehat{\rho})$ and any χ such that

$$(w;\widehat{\rho}) \models_{\chi} \big(\beta;(\sigma;\rho)\big), \tag{20}$$

so that

$$\mathcal{V}^I_{(w;\widehat{\rho})/\chi}[F] = \textbf{true} \tag{21}$$

and $(w;\widehat{\rho}) \sqsubseteq (\sigma;\rho)$, and hence

$$w \sqsubseteq \sigma \tag{22}$$

and

$$\widehat{\rho} \supseteq \rho. \tag{23}$$

We need to prove that there is some $z \in \{1,\ldots,m\}$ such that $(w;\widehat{\rho}) \sqsubseteq (\sigma_z;\rho_z)$. Let

$$\rho' = \widehat{\rho} \restriction \big[Dom(\rho_1) \cup \cdots \cup Dom(\rho_m) \cup CS(F)\big].$$

Since $\widehat{\rho} \supseteq \rho$ and $\rho_i \supseteq \rho$ for every i, we conclude $\rho' \supseteq \rho$, which means that we must have $\rho' = \rho_i'$ for some $i \in \{1,\ldots,d\}$. Further, let $\chi_j = \chi \restriction FV(F)$. By (21), Lemma 3, and Lemma 5, we infer

$$\mathcal{V}^I_{(w;\rho_i')/\chi_j}[F] = \textbf{true},$$

and hence, by C', $(w;\rho_i') \sqsubseteq (\sigma_z;\rho_z)$ for some $z \in \{1,\ldots,m\}$. Since $\rho_i' = \rho'$, that entails $\rho' \supseteq \rho_z$, and since $\widehat{\rho} \supseteq \rho'$, we infer $\widehat{\rho} \supseteq \rho_z$ and hence $(w;\widehat{\rho}) \sqsubseteq (\sigma_z;\rho_z)$. □

Although the above result is important for mechanizing the semantics of Vivid, its significance is primarily theoretical. In most cases it would be infeasible to decide entailment by carrying out the above procedure. The principal difficulty is that the procedure requires iteration over all worlds $w \sqsubseteq \sigma$, and when the number of such worlds is very large (e.g., 10^{10} or higher), such iteration would be wildly impractical. We have developed methods that can often decide the entailment relation much more efficiently, capable of providing results even in some cases where the number of worlds is astronomically large. There is not enough space in this paper to describe these developments in detail, but a flavor will be given in Section 7.

6. A family of diagrammatic natural deduction languages

We now formally define Vivid, a family of natural deduction languages in the DPL vein [4] 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 Σ 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 as described in Section 5. We write $F[t/v]$ to denote the formula obtained from F by replacing every free occurrence of v by the term t (taking care to rename F if necessary to avoid variable capture). The following result is proved by induction on the structure of F.

Lemma 20. *If* $b \in \{\textbf{true}, \textbf{false}\}$,

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

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

$$\mathcal{V}^I_{(w;\rho)/\chi[v'\mapsto s]}\big[F[v'/v]\big] = b.$$

6.1. Abstract syntax

There are two syntactic categories of proofs, sentential and diagrammatic. Sentential deductions are used to derive formulas, while diagrammatic deductions are used to derive diagrams. We will see that the two can be freely mixed, and indeed that their structures are mutually recursive. We use the letters D and Δ to range over sentential and diagrammatic deductions, respectively. The symbol $\mathfrak{D}$ will range over the union of the two. The abstract syntax [46] of both proof types is defined by the grammars below:

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

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

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

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

RuleApp ::=
- **claim** F
- | **true-intro**
- | **false-elim**
- | **modus-ponens** $F \Rightarrow G, F$
- | **modus-tollens** $F \Rightarrow G, \neg G$
- | **double-negation** $\neg\neg F$
- | **absurd** $F, \neg F$
- | **left-and** $F \wedge G$
- | **right-and** $F \wedge G$
- | **both** F, G
- | **left-either** F, G
- | **right-either** F, G
- | **cases** $F_1 \vee F_2, F_1 \Rightarrow G, F_2 \Rightarrow G$
- | **left-iff** $F \Leftrightarrow G$
- | **right-iff** $F \Leftrightarrow G$
- | **equiv** $F \Rightarrow G, G \Rightarrow F$

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

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

We omit the defining equations for the sentential **pick-witness**, which is handled like the diagrammatic **pick-witness**; and for the remaining **cases by**, which is treated like the one above. The definition for the other forms is straightforward and can be found elsewhere [4]. In all cases we assume that the deduction has been α-renamed away from the given term t.

6.2. Evaluation semantics

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

$$\gamma \vdash D \rightsquigarrow F$$

and

$$\gamma \vdash \Delta \rightsquigarrow (\sigma; \rho),$$

which are read as follows:

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

We refer to rules that derive judgments of the former sort as "sentential," since they prescribe the behavior of sentential deductions; while rules that derive the second sort of judgment are called diagrammatic, since they prescribe the operational meaning of diagrammatic deductions.

The semantics of most sentential deductions are straightforward generalizations of the standard DPL semantics for natural deduction [4]. We illustrate here with the axiom for **left-and** and the rule for **assume**, omitting the rest:

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

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

In addition, for convenience, we introduce the sentential form

$$F \textbf{ by absurdity} \tag{24}$$

as syntax sugar for the following sentential deduction:

assume $\neg F$
  claim false; // This gives $\neg F \Rightarrow$ **false**, provided that **false** is in the assumption base
false-elim; // This gives $\neg$**false**
modus-tollens $\neg F \Rightarrow$ **false**, $\neg$**false**; // This gives $\neg\neg F$
double-negation $\neg\neg F$ // Finally, this produces F

Accordingly, this desugaring ensures that the semantics of (24) are as follows:

$$\overline{(\beta \cup \{\textbf{false}\}; (\sigma; \rho)) \vdash F \textbf{ by absurdity } \rightsquigarrow F}$$

Some additional syntax sugar is introduced later (p. 1395).

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

$$\frac{}{(\beta; (\sigma; \rho)) \vdash \textbf{observe } F \rightsquigarrow F} \quad [\textit{Observe}]$$
$$\text{provided that } I_{(\sigma;\rho)/\chi}(F) = \textbf{true} \text{ for all } \chi$$

This rule is used to extract sentential information from diagrams. The side condition is computable because of Lemma 7 and because, by Lemma 5, we only need to be concerned with the free variables of F. In fact usually F is a sentence (it has no free variables), and hence we only need to consider the empty variable assignment $\emptyset$.

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 follows 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 standard symbolic logic unchanged, since there is no diagram manipulation involved:

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

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

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

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 Fig. 4.

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

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

This is classified as a sentential deduction, since the end result is a formula F. Its semantics are shown in Fig. 5. The caveat that the diagrams $(\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)$ form an exhaustive set of possibilities on the basis of $F_1, \ldots, F_k$ and the current diagram is formally captured by the proviso

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

When $k = 0$, the **by** keyword is omitted, and we simply write

$$\textbf{cases}\colon (\sigma_1; \rho_1) \rightarrow D_1 \mid \cdots \mid (\sigma_n; \rho_n) \rightarrow D_n.$$

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

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

The corresponding semantics are given by rule $[C_1]$, shown in Fig. 4. Again, we might have $k = 0$, and in that case the **by** keyword is omitted.

Likewise, there are four types of deduction sequencing:

1. $D_1; D_2$, where a sentential deduction D_1 is composed with another sentential deduction D_2. This form is classified as a sentential deduction, since the end result is a formula (the conclusion of D_2). Its semantics are given by rule $[D; D]$ of Fig. 4. They are isomorphic to the regular composition semantics of DPLs, since there is no diagram manipulation involved.

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

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

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

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

$$\frac{\begin{array}{c}(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_1; \rho_1)) \vdash \Delta_1 \rightsquigarrow (\sigma'; \rho') \\ \vdots \\ (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_n; \rho_n)) \vdash \Delta_n \rightsquigarrow (\sigma'; \rho')\end{array}}{\begin{array}{c}(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \vdash \textbf{cases by } F_1, \ldots, F_k\colon\ (\sigma_1; \rho_1) \rightarrow \Delta_1 \mid \cdots \mid (\sigma_n; \rho_n) \rightarrow \Delta_n \rightsquigarrow (\sigma'; \rho') \\ \text{provided } (\sigma; \rho) \Vdash_{\{F_1, \ldots, F_k\}} \{(\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)\}\end{array}} \quad [C_1]$$

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

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

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

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

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

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

Fig. 4. Formal semantics of diagrammatic deductions.

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

$$\frac{\begin{array}{c}(\beta \cup \{F_1, \ldots, F_k\}; (\sigma_1; \rho_1)) \vdash D_1 \rightsquigarrow F \\ \vdots \\ (\beta \cup \{F_1, \ldots, F_k\}; (\sigma_n; \rho_n)) \vdash D_n \rightsquigarrow F\end{array}}{\begin{array}{l}(\beta \cup \{F_1, \ldots, F_k\}; (\sigma; \rho)) \vdash \textbf{cases by } F_1, \ldots, F_k\text{: } (\sigma_1; \rho_1) \rightarrow D_1 \mid \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdots \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad (\sigma_n; \rho_n) \rightarrow D_n \rightsquigarrow F \\ \qquad \text{provided } (\sigma; \rho) \Vdash_{\{F_1, \ldots, F_k\}} \{(\sigma_1; \rho_1), \ldots, (\sigma_n; \rho_n)\}\end{array}} \; [C_3]$$

Fig. 5. Semantics of diagrammatic-to-sentential case reasoning.

4. $\Delta_1; \Delta_2$, where a diagrammatic deduction Δ_1 is composed with another diagrammatic deduction Δ_2. This form is classified as a diagrammatic deduction, since the end result is a diagram (the conclusion 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.

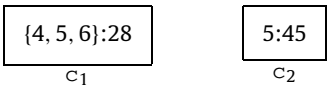
We briefly discuss the remaining rules of Fig. 4. [*Thinning*] is probably the most frequently used rule for heterogeneous inference in Vivid. It allows us to refine the current state by ruling out worlds that are inconsistent with the cited formulas. [*Widening*] can be seen as the inverse of thinning, entitling us to "lose information" by increasing rather than decreasing the number of possible worlds that satisfy the current state. This can be useful in getting the diagrammatic branches of a case analysis to be identical. [*Absurdity*] entitles us to infer any diagram whatsoever from a contradiction. [*Diagram-Reiteration*] allows us to retrieve the current diagram. [*EI*/Δ] is a diagrammatic version of existential instantiation, whereby we unpack an existential quantification by choosing a witness and then proceed with a diagrammatic deduction.

Theorem 21 *(Soundness). If $\gamma \vdash D \rightsquigarrow F$ then $\gamma \models F$; and if $\gamma \vdash \Delta \rightsquigarrow (\sigma; \rho)$ then $\gamma \models (\sigma; \rho)$.*

Theorem 22 *(Computability). The proof-checking problem for Vivid is decidable. That is, there is an algorithm that will take any proof $\mathfrak{D}$, context $(\beta; (\sigma; \rho))$, and result r (proposition or diagram), and will determine whether or not $(\beta; (\sigma; \rho)) \vdash \mathfrak{D} \rightsquigarrow r$.*

Proof. A straightforward induction on the structure of $\mathfrak{D}$ will provide a recursive definition of the requisite proof-checking algorithm. An inspection of the evaluation rules of the language will confirm that all of them are computable. The only interesting cases are rules with side conditions involving either the function $I_{(\sigma;\rho)/\chi}$ or the entailment relation $\Vdash$, both of which are computable (by Theorem 7 and Theorem 19, respectively). □

Example 10. Consider the Vivid language obtained by fixing the clock signature, attribute structure and interpretation of Example 8. Now consider a system of two clocks s_1 and s_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.[26]



Intuitively, this state signifies that we know the precise time displayed by s_2 (5:45 am). We are also sure of the minute value of s_1 (28), but not of its hour value, which could be either 4, 5, or 6. Now suppose that we are further given the premise $\mathtt{Ahead}(\mathtt{c}_1, \mathtt{c}_2)$, indicating that the time displayed by s_1 is ahead of that displayed by s_2.

From these two pieces of information, one diagrammatic and the other sentential, we should be able to infer the following diagram, call it σ' (which is, in fact, a world):

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

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

[26] An exceedingly simple picture, but a picture nonetheless. This example is used here only because its simplicity makes it ideal for exposition purposes, not because it provides a compelling demonstration of the utility of diagrams in problem solving.

This deduction, when evaluated in a context of the form $(\beta \cup \{\texttt{Ahead}(\texttt{c}_1, \texttt{c}_2)\}; (\sigma; \rho))$, will result in the state (diagram) $(\sigma'; \rho)$. More formally, we have:

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

by virtue of

$$(\sigma; \rho) \Vdash_{\{\texttt{Ahead}(\texttt{c}_1, \texttt{c}_2)\}} (\sigma'; \rho). \tag{25}$$

Note that ρ does not change in the resulting state.

To establish (25) rigorously, we must show that for all worlds $(w; \widehat{\rho})$ and variable assignments χ, if

$$(w; \widehat{\rho}) \models_\chi \big(\big\{\texttt{Ahead}(\texttt{c}_1, \texttt{c}_2)\big\}; (\sigma; \rho)\big)$$

then $(w; \widehat{\rho}) \models (\sigma'; \rho)$. That is, intuitively, if any world $(w; \widehat{\rho})$ consistent with the current state $(\sigma; \rho)$—i.e., such that $(w; \widehat{\rho}) \sqsubseteq (\sigma; \rho)$—satisfies the formula $\texttt{Ahead}(\texttt{c}_1, \texttt{c}_2)$, w.r.t. any χ, then that world must be consistent with $(\sigma'; \rho)$, i.e., we must have $(w; \widehat{\rho}) \sqsubseteq (\sigma'; \rho)$. To that end, pick any $(w; \widehat{\rho}) \sqsubseteq (\sigma; \rho)$ and any χ such that

$$\mathcal{V}^I_{(w;\widehat{\rho})/\chi}\big[\texttt{Ahead}(\texttt{c}_1, \texttt{c}_2)\big] = \textbf{true}. \tag{26}$$

There are, in fact, only three worlds $w \sqsubseteq \sigma$:

$$w_1\colon \quad \mathit{hours}(s_1) = \{4\}, \quad \mathit{minutes}(s_1) = \{28\}, \quad \mathit{hours}(s_2) = \{5\}, \quad \mathit{minutes}(s_1) = \{45\};$$
$$w_2\colon \quad \mathit{hours}(s_1) = \{5\}, \quad \mathit{minutes}(s_1) = \{28\}, \quad \mathit{hours}(s_2) = \{5\}, \quad \mathit{minutes}(s_1) = \{45\};$$
$$w_3\colon \quad \mathit{hours}(s_1) = \{6\}, \quad \mathit{minutes}(s_1) = \{28\}, \quad \mathit{hours}(s_2) = \{5\}, \quad \mathit{minutes}(s_1) = \{45\}.$$

Now (26) does not hold for $w = w_1$ and $w = w_2$, so we do not need to consider these two worlds. For the third and last possibility, we do have

$$(w_3; \widehat{\rho}) \models_\chi \big(\big\{\texttt{Ahead}(\texttt{c}_1, \texttt{c}_2)\big\}; (\sigma; \rho)\big)$$

(again, for arbitrary $\widehat{\rho} \supseteq \rho$ and χ), and the result follows because $(w_3; \widehat{\rho}) \sqsubseteq (\sigma'; \rho)$.

7. Implementation

A quick inspection of the formal semantics of Vivid, in particular the [*Observe*] rule (p. 1386) and the rules for the remaining constructs (Figs. 4 and 5) reveals that there are two operations that must be implemented in order to mechanize these semantics:

1. evaluating a formula in a given named state and variable assignment; and
2. deciding the entailment relation of Definition 7.

From these two, the second is much more likely to become a bottleneck, since it may need to invoke the first operation (formula evaluation) a very large number of times. A naive implementation of these operations in terms of the definitions and results given so far (specifically, in terms of Definition 11 and Theorem 19) would be unacceptably inefficient. This is primarily because such an implementation might require, in the worst case, iteration over all worlds extending the given state. While such iteration may be feasible in some cases, in many other cases it will be practically impossible, as the given state will simply be too large.

There are several avenues for arriving at more efficient implementations. One of them is to translate both diagrams and Vivid formulas into propositional-logic sentences in conjunctive normal form, model both problems as satisfiability questions, and tackle them with off-the-shelf SAT solvers. The translation would be fairly straightforward, as a state is essentially a CNF sentence asserting that the value of the first attribute for the first object is v_1 *or* v_2 *or* ...; *and* the value of the second attribute for the first object is v'_1 *or* v'_2 *or* ...; and so on. For instance, let σ be the following state of a system comprising two objects s_1 and s_2, and two attributes, *color* and *size*:

σ
$\mathit{color}(s_1) = \{\mathit{Red}, \mathit{Blue}\}$
$\mathit{size}(s_1) = \{\mathit{Small}, \mathit{Medium}, \mathit{Large}\}$
$\mathit{color}(s_2) = \{\mathit{Red}, \mathit{Blue}, \mathit{Green}\}$
$\mathit{size}(s_2) = \{\mathit{Medium}, \mathit{Large}\}$

This state can be represented by the CNF sentence:

$$(\mathit{Red}_1 \vee \mathit{Blue}_1) \wedge (\mathit{Small}_1 \vee \mathit{Medium}_1 \vee \mathit{Large}_1) \wedge (\mathit{Red}_2 \vee \mathit{Blue}_2 \vee \mathit{Green}_2) \wedge (\mathit{Medium}_2 \vee \mathit{Large}_2).$$

The number of boolean variables in such a straightforward translation would be linear in the sizes of the ascriptions and the number of system objects. The translations for named states would need to be more complicated to account for constant assignments. Translations of Vivid formulas would have to be carried out on an individual basis, in accordance with the definitions of the various relations in the underlying attribute structure. Formula evaluation would then become tantamount to implication. Countermodels found by the SAT solver would be translated back to Vivid states and displayed diagrammatically. We have implemented such SAT-based instances of Vivid and found their performance to be adequate for most practical purposes. Alternatively, OBDDs could be used to represent both system states and Vivid formulas, although we have not had any practical experience with such implementations.

While such approaches are fairly straightforward and might often result in acceptable implementations, we observe that the problems in question have a good deal of structure that we can sometimes exploit to arrive at efficient solutions even in cases well beyond the reach of the current state of the art of SAT solvers or OBDDs, namely, cases with many thousands of boolean variables and millions of clauses. That is not always achievable, of course, and in some cases the specialized techniques we have developed are outperformed by more straightforward implementations relying on generic technologies. Fortunately, it is easy to determine ahead of time whether or not the custom-made techniques will be profitable by performing a few simple calculations. The appropriate implementation technology can then be selected on the basis of those results. We do not have enough space here to present these techniques, most of which are based on treating states as languages, and specifically as acyclic deterministic finite automata (ADFA); more details can be found in the longer online report. Here we will only give a glimpse of how the first problem can be tackled—evaluating a formula in a given state—without explicitly carrying out the evaluation in every world of that state, as a naive reading of Definition 11 would suggest.

Lemma 6 along with Lemmas 8 through 12 already point to the main idea for short-circuited formula evaluation that avoids such iteration, but the caveats of the conjunction and disjunction lemmas (and the similar caveats for universal and existential quantifications) mean that an evaluation technique based directly on those lemmas would not be universally applicable. For instance, if we are evaluating a conjunction $F_1 \wedge F_2$ and the short-circuited evaluation of both F_1 and F_2 returns **unknown**, there is no specific value we can infer for the conjunction; the result could be either **false** or **unknown**. In such cases we would need to revert to exhaustive formula evaluation in every world of the given state. The simple idea below can avoid this problem in many cases.

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

$$\mathcal{B}(\textbf{true}, \rho, \chi) = \emptyset;$$

$$\mathcal{B}(\textbf{false}, \rho, \chi) = \emptyset;$$

$$\mathcal{B}\big(R(t_1, \ldots, t_n), \rho, \chi\big) = \begin{cases} \{(l_{i_1}; t_{j_1}^{\rho,\chi}), \ldots, (l_{i_m}; t_{j_m}^{\rho,\chi})\} & \text{if } t_{j_i}^{\rho,\chi} \downarrow \text{ for every } i \in \{1, \ldots, m\}; \\ \infty & \text{otherwise;} \end{cases}$$

$$\mathcal{B}(\neg F, \rho, \chi) = \mathcal{B}(F, \rho, \chi);$$

$$\mathcal{B}(F_1 \wedge F_2, \rho, \chi) = \mathcal{B}(F_1, \rho, \chi) \cup \mathcal{B}(F_2, \rho, \chi);$$

$$\mathcal{B}(F_1 \vee F_2, \rho, \chi) = \mathcal{B}(F_1, \rho, \chi) \cup \mathcal{B}(F_2, \rho, \chi);$$

$$\mathcal{B}(\forall v \,.\, F, \rho, \chi) = \bigcup_{i=1}^{n} \mathcal{B}\big(F, \rho, \chi[v \mapsto s_i]\big);$$

$$\mathcal{B}(\exists v \,.\, F, \rho, \chi) = \bigcup_{i=1}^{n} \mathcal{B}\big(F, \rho, \chi[v \mapsto s_i]\big).$$

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

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

$$L_1 \equiv_S L_2 \Leftrightarrow \big[\forall x \in S \,.\, L_1(x) = L_2(x)\big]. \tag{27}$$

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

$$w_1 \equiv_S w_2 \Leftrightarrow \big[\forall (l; s) \in S \,.\, w_1(l, s) = w_2(l, s)\big]. \tag{28}$$

If we think of worlds as strings, then the above definition is isomorphic to (27), with a.o. pairs playing the role of string positions. The following is a trivial consequence of the above definition, but useful enough to isolate as a lemma:

Lemma 23. *If $w_1 \equiv_S w_2$ then $w_1 \equiv_{S'} w_2$ for every $S' \subseteq S$.*

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

Theorem 24. *If* $\mathcal{B}(F, \rho, \chi) \neq \infty$ *and* $w_1 \equiv_{\mathcal{B}(F,\rho,\chi)} w_2$, *then* $\mathcal{V}^I_{(w_1;\rho)/\chi}[F] = \mathcal{V}^I_{(w_2;\rho)/\chi}[F]$.

To illustrate the utility of the above result, consider again the system of the two clocks s_1 and s_2 of Example 9, and let σ be the following state of that system:

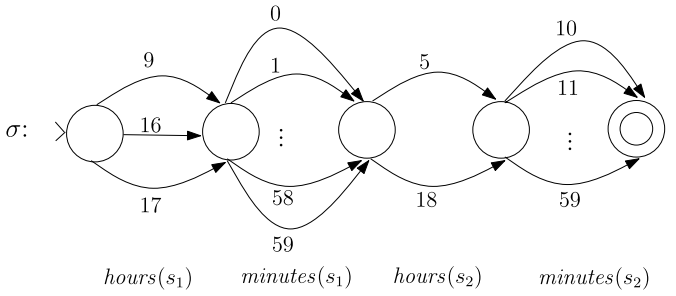
$$\sigma(\mathit{hours}, s_1) = \{9, 16, 17\};$$

$$\sigma(\mathit{minutes}, s_1) = \{0, 1, 2, \ldots, 59\};$$

$$\sigma(\mathit{hours}, s_2) = \{5, 18\};$$

$$\sigma(\mathit{minutes}, s_2) = \{10, 11, \ldots, 59\}.$$

Viewed as an ADFA, σ can be depicted as follows:



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

$$I_{(\sigma;\rho)/\chi}\big(\forall x\,.\,\mathtt{PM}(x)\big) = \textbf{unknown} \tag{29}$$

for arbitrary ρ and χ. However, if we simply tried to compute $I_{\rho/\chi[x\mapsto s_1]}(\mathtt{PM}(x))$ and $I_{\rho/\chi[x\mapsto s_2]}(\mathtt{PM}(x))$, we would obtain **unknown** and **unknown**, respectively, and therefore, by part (b) of Lemma 11, we would not be able to infer (29), because it would not be possible to rule out the possibility $I_{(\sigma;\rho)/\chi}(\forall x\,.\,\mathtt{PM}(x)) = \textbf{false}$. Accordingly, we would need to resort to calculating

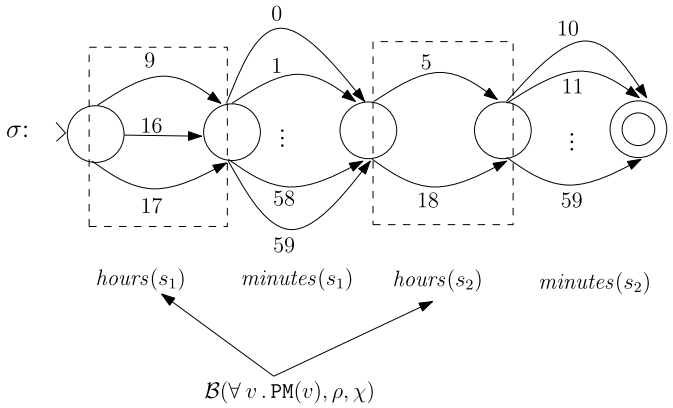
$$\mathcal{V}^I_{(w;\rho)/\chi}\big[\forall x\,.\,\mathtt{PM}(x)\big]$$

for every $w \sqsubseteq \sigma$. But that could be very inefficient. Depending on how we iterate through the space of 18,000 worlds, we might have to evaluate $\forall x\,.\,\mathtt{PM}(x)$ over one hundred times before we find two worlds w_i and w_j such that

$$\mathcal{V}^I_{(w_i;\rho)/\chi}\big[\forall x\,.\,\mathtt{PM}(x)\big] = \textbf{true} \quad \text{and} \quad \mathcal{V}^I_{(w_j;\rho)/\chi}\big[\forall x\,.\,\mathtt{PM}(x)\big] = \textbf{false},$$

which would allow us to arrive at the answer (29).

But examining all these worlds is unnecessary for a simple reason: The *minutes* attribute of a clock is irrelevant in determining whether or not the clock's time is p.m. Only the value of *hours* is necessary for making that judgment. If that value is greater than 11, then the clock is p.m., otherwise it is not. By taking advantage of this information, which is provided by the profile of PM, we can restrict attention only to the *hours* attribute of the two clocks. That is precisely what the basis of F enables us to do:



Arbitrary values can be selected for the *minutes* attribute, say 0 for $minutes(s_1)$ and 10 for $minutes(s_2)$. We can then narrow down the relevant space from 18,000 worlds to only 6 worlds $w_1, \ldots, w_6$, formed by choosing a value for $hours(s_1)$ (three choices) followed by a value for $hours(s_2)$ (two choices), and keeping $minutes(s_1)$ and $minutes(s_2)$ fixed to 0 and 10, respectively:

$$
\begin{array}{llll}
w_1(hours, s_1) = 9 & w_1(minutes, s_1) = 0 & w_1(hours, s_2) = 5 & w_1(minutes, s_2) = 10; \\
w_2(hours, s_1) = 9 & w_2(minutes, s_1) = 0 & w_2(hours, s_2) = 18 & w_2(minutes, s_2) = 10; \\
w_3(hours, s_1) = 16 & w_3(minutes, s_1) = 0 & w_3(hours, s_2) = 5 & w_3(minutes, s_2) = 10; \\
w_4(hours, s_1) = 16 & w_4(minutes, s_1) = 0 & w_4(hours, s_2) = 18 & w_4(minutes, s_2) = 10; \\
w_5(hours, s_1) = 17 & w_5(minutes, s_1) = 0 & w_5(hours, s_2) = 5 & w_5(minutes, s_2) = 10; \\
w_6(hours, s_1) = 17 & w_6(minutes, s_1) = 0 & w_6(hours, s_2) = 18 & w_6(minutes, s_2) = 10.
\end{array}
$$

Now $\forall x\,.\,\mathtt{PM}(x)$ will only need to be evaluated in four worlds (w_1–w_4) before we determine the correct answer, (29).

Computing $\mathcal{B}(F, \rho, \chi)$ allows us to perform this type of narrowing in a systematic and sound way. Roughly, atomic formulas $R(t_1, \ldots, t_n)$ are restricted to appropriate a.o. pairs by utilizing the profile information of R, while the basis of a complex formula is computed recursively (and conservatively) by joining the bases of each subformula. Since computing the basis of a formula is very cheap (linear both in time and space), we can statically analyze every formula F that needs to be evaluated in a given state σ and assignments ρ and χ by computing $\mathcal{B}(F, \rho, \chi)$, picking arbitrary values for the a.o. pairs which are *not* in the basis, and constructing all and only those worlds w' which reflect different combinations of values for a.o. pairs in the basis. Theorem 24 then allows us to evaluate F only w.r.t. to such worlds w'.

8. Seating puzzles

We now consider a puzzle that has become somewhat of a classic in discussions of diagrammatic reasoning. Presented (and to the best of our knowledge, devised) by Barwise and Etchemendy [6], it is typical of the sort of problems found in the analytical section of the GRE, as well as typical of cognitive science experiments investigating spatial reasoning [14].[27] The puzzle can be described as follows:

> Five people *A*, *B*, *C*, *D*, and *E* are to be seated in a row of five seats. The seating arrangement must satisfy the following three conditions:
> 1. *A* and *C* should flank *E*.
> 2. *C* should be closer to the middle seat than *B*.
> 3. *B* and *D* should be seated next to each other.
>
> On the basis of this information:
> (a) Prove that *E* cannot be either in the middle or on either end.
> (b) Can it be determined who must be seated in the middle seat?
> (c) Can it be determined who is to be seated on the two ends?

Looking ahead, all three questions are answered with one Vivid deduction, shown in Fig. 6. Note that the Vivid system we are about to present is not hardwired to this particular puzzle, but allows for the formulation and solution of a wide variety of seating puzzles (involving arbitrarily many seats, persons, names thereof, and combinations of constraints). We have also defined and implemented extensions of the system to two dimensions, involving several rows of seats and relations such as *above* and *below*.

[27] The puzzle is also discussed by Shin [52], by Barwise and Etchemendy [9], and others.

```
cases by flanking(A, C, E), distinct-seats :
  A E C ? ?   →  observe goal
| ? A E C ?   →  begin
                   observe ¬adjacent(B, D);
                   absurd adjacent(B, D), ¬adjacent(B, D);
                   goal by absurdity
                 end
| ? ? A E C   →  begin
                   observe ¬closerToCenter(C, B);
                   absurd closerToCenter(C, B), ¬closerToCenter(C, B);
                   goal by absurdity
                 end
| C E A ? ?   →  begin
                   observe ¬closerToCenter(C, B);
                   absurd closerToCenter(C, B), ¬closerToCenter(C, B);
                   goal by absurdity
                 end
| ? C E A ?   →  begin
                   observe ¬adjacent(B, D);
                   absurd adjacent(B, D), ¬adjacent(B, D);
                   goal by absurdity
                 end
| ? ? C E A   →  observe goal
```

Fig. 6. A Vivid deduction solving the seating puzzle of Barwise and Etchemendy [6].

If we make the convention that juxtaposition indicates seating adjacency, and that the left-to-right relation in the text corresponds to the analogous relation in the row of seats, then a seating arrangement might be graphically depicted simply as follows:

$$A\ E\ C\ B\ D \tag{30}$$

Note that this particular arrangement satisfies all three constraints. We will also make the natural convention of identifying the five chairs from left to right with the five integers $1, \ldots, 5$, so that the leftmost chair is chair 1 and the rightmost is chair 5. A questionmark over a seat indicates that we do not know who (if anyone) is to be placed in that seat. Thus, for instance, the "diagram"

$$?\ ?\ A\ ?\ ? \tag{31}$$

indicates a seating arrangement in which A is seated in chair 3, but we do not know where the others are seated (other than the fact that they are not in the middle).

We now introduce a simple Vivid language that lets us solve problems in this domain. Intuitively, the system objects will be persons, or more abstractly, objects to be seated; and their attribute values will be seat numbers, or more precisely, sets of seat numbers, allowing for incomplete information. Accordingly, a system state will map each such object to a set of positive integers representing chairs. For instance, diagram (30) corresponds to the system state

$$seat(A) = 1, \quad seat(E) = 2, \quad seat(C) = 3, \quad seat(B) = 4, \quad seat(D) = 5,$$

which is a world, while diagram (31) is captured by the system state:

$$seat(A) = 3, \quad seat(E) = \{1, 2, 4, 5\}, \quad seat(C) = \{1, 2, 4, 5\}, \quad seat(B) = \{1, 2, 4, 5\}, \quad seat(D) = \{1, 2, 4, 5\}. \tag{32}$$

The vocabulary for this instance of Vivid consists of the binary symbol for identity ($=$), along with six relation symbols, whose intuitive semantics are as follows:

1. **flanking**$(x, y, z) \equiv x$ and y are flanking z.
2. **adjacent**$(x, y) \equiv x$ and y are seated next to each other.
3. **atEnd**$(x) \equiv x$ is seated at an end seat (far left or far right).
4. **middle**$(x) \equiv x$ is seated in the middle seat.
5. **closerToCenter**$(x, y) \equiv x$ is seated closer to the middle seat than y.
6. **sameSeat**$(x, y) \equiv x$ is assigned the same seat as y. (We will show that this holds iff $x = y$.)

We let $x, y, z, \ldots$, possibly with subscripts, serve as variables. The constant names are $A, B, C, D, E, \ldots$, also possibly with subscripts. The attribute structure here is parameterized over the number of seats, $m > 1$:

$$\mathcal{A}_{S_m} = \big(seat\colon \{1, \ldots, m\}; \{R_1, R_2, R_3, R_4, R_5, R_6\}\big), \quad \text{with:}$$

1. $R_1 \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\} \times \{1, \ldots, m\}$, defined as follows: $R_1(i, j, k) \Leftrightarrow |i - k| = |j - k| = 1 \wedge i \neq j$. This interprets **flanking** via the profile $[(seat, 1), (seat, 2), (seat, 3)]$.
2. $R_2 \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$, defined as $R_2(i, j) \Leftrightarrow |i - j| = 1$. This interprets **adjacent** via the profile $[(seat, 1), (seat, 2)]$.
3. $R_3 \subseteq \{1, \ldots, m\}$, defined as $R_3(i) \Leftrightarrow i = 1 \vee i = m$. This interprets **atEnd** via the profile $[(seat, 1)]$.
4. $R_4 \subseteq \{1, \ldots, m\}$, defined as $R_4(i) \Leftrightarrow i = \lceil \frac{m}{2} \rceil$. This interprets **middle** via the profile $[(seat, 1)]$.
5. $R_5 \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$, defined as

$$R_5(i, j) \Leftrightarrow \left| i - \left\lceil \frac{m+1}{2} \right\rceil \right| < \left| j - \left\lceil \frac{m+1}{2} \right\rceil \right|.$$

   This interprets **closerToCenter** via the profile $[(seat, 1), (seat, 2)]$.
6. $R_6 \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$, defined as $R_5(i, j) \Leftrightarrow i = j$. This interprets **sameSeat** via the profile

$$\big[(seat, 1), (seat, 2)\big].$$

The identity symbol is interpreted by the identity relation on the set of objects.[28]

An obvious constraint that we might wish to enforce is that distinct objects must be placed in distinct seats. This will be captured by the sentence

$$\forall x, y\ .\ \mathbf{sameSeat}(x, y) \Rightarrow x = y.$$

For brevity, we will refer to this formula as *distinct-seats*.

Even though diagrams in this system are particularly simple, being one-dimensional and consisting of plain text, a few remarks on how to parse them are in order. This is a domain in which every system object needs to be labeled in order to be uniquely identifiable, because without such labeling the identity of a system object cannot be inferred from a diagram. The reason is that an object here does not occupy a fixed place in every diagram across a given proof,[29] in contrast, say, with the map-coloring puzzles of the next section, where a system object (a map region) has a unique diagrammatic location throughout a proof. Accordingly, in an implementation of this system the user is required to fix two parameters at the beginning of a session (i.e., prior to entering and evaluating a proof): the number of seats m, and the objects to be seated, specified simply as distinct names $c_1, \ldots, c_n$. (Note that the number of objects to be seated, n, cannot exceed the number of seats m, although we could have $n < m$.) For instance, in the case of the particular puzzle discussed in the beginning, the user would specify five chairs and the five objects A, B, C, D, and E. This lets the implementation know that every system state thereafter would consist of n objects $s_1, \ldots, s_n$, named throughout by the constant assignment $\rho = \{c_1 \mapsto s_1, \ldots, c_n \mapsto s_n\}$. Of course this constant assignment could be subsequently extended, if part of the problem involves determining the referents of certain additional names. For example, in the opening puzzle, the second question could be expressed by imposing the constraint **middle**(G), where the identity of G is initially unknown; the question could then be answered by proving $G = C$. If a system object s_i does not appear in a diagram (or more precisely, if the corresponding name c_i does not), then $seat(s_i)$ is defined as the set of all seats that are free in that diagram, where a free seat is indicated by the presence of a questionmark. For instance, B, C, D, and E do not appear in (31), and therefore in the system state (32) obtained by parsing (31), the *seat* ascription maps the corresponding objects to the set of all chairs that are free in that diagram, namely, $\{1, 2, 4, 5\}$. With these conventions in place, the implementation can readily parse any given diagram (a sequence of m tokens, each of which is either a questionmark or has exactly one of the n names placed on it, and such that no name appears more than once) into a unique named state.

We now discuss the solution presented in Fig. 6. That deduction, when evaluated in a context comprising the minimal-information diagram

? ? ? ? ?

and an assumption base that contains the *distinct-seats* postulate along with the three given constraints (formalized in this language as **flanking**(A, C, E), **closerToCenter**(C, B), and **adjacent**(B, D)), will derive the conclusion

$$\neg\mathbf{atEnd}(E) \wedge \neg\mathbf{middle}(E) \wedge \mathbf{middle}(C),$$

which we abbreviate as *goal*. The Vivid deduction is a faithful representation of the informal reasoning that a human problem-solver would use when tackling the puzzle, which is roughly as follows. Knowing that E must be between A and C, we can distinguish three main cases:

[28] Strictly speaking, we should make the attribute structure automorphic if we are to formalize the realization of the identity relation symbol via a profile, but we ignore this minor technicality here in the interest of simplicity.

[29] For instance, an object might not appear at all in the seating arrangements during the first few steps of a proof, when we do not yet have enough information to place it correctly, but might appear in the diagram subsequently, once such information is deduced.

1. *A E C ? ?*
2. *? A E C ?*
3. *? ? A E C*

By symmetry, there are three mirror cases, obtained from the above by flipping the positions of A and C, for a total of six cases, as shown in Fig. 6. These six cases represent the only possibilities consistent with the flanking constraint which dictates that E must be between A and C, assuming that distinct persons are assigned distinct seats. Moreover, from these six cases, only

$$A\ E\ C\ ?\ ? \tag{33}$$

and

$$?\ ?\ C\ E\ A \tag{34}$$

are consistent with the other two constraints. Each of the remaining four cases either makes it impossible for B and D to be adjacent or else fails to place C closer to the middle than B. So in those cases *goal* follows trivially by contradiction, while in cases (33) and (34) *goal* follows by simple inspection (by applying **observe**). The derivation of *goal* explicitly answers the first two questions of the puzzle. For the third question, a simple glance at the proof reveals that in the only two cases that do not involve a contradiction (namely, the top and bottom clauses in Fig. 6, i.e., cases (33) and (34)), there are no two unique persons who must be seated on the two ends; any one of A, B, and D could appear at an end.[30]

As we pointed out, four of the six alternatives in the above case analysis are ruled out because they violate certain constraints. Hence, in these four cases, the goal follows vacuously: We observe that the constraint does not hold, we derive a contradiction from the fact that it ought to hold, and then we derive the goal by absurdity. This situation arises frequently in Vivid, so it is helpful to have some syntax sugar that routinely fills in these steps without the need to spell them out. This is accomplished by the "**violates** F" construct, which may appear immediately following an arrow -> in a branch of a case analysis:

$$\textbf{cases by}\ F_1 \cdots F_k : (\sigma_1; \rho_1) \to \mathfrak{D}_1 \mid \cdots \mid (\sigma_i; \rho_i) \to \textbf{violates}\ F \mid \cdots \mid (\sigma_l; \rho_l) \to \mathfrak{D}_l. \tag{35}$$

If F is in the assumption base, and if it comes out false under $(\sigma_i; \rho_i)$ and the current assumption base, the phrase **violates** F will derive, by way of contradiction, the conclusion that is derived by the other tail deductions $\mathfrak{D}_j$.[31] Using this convenient syntax sugar, the solution can be expressed more succinctly as follows:

cases by flanking(A, C, E), *distinct-seats*:
A E C ? ? → **observe** *goal*
? A E C ? → **violates adjacent**(B, D)
? ? A E C → **violates closer-to-center**(C, B)
C E A ? ? → **violates closer-to-center**(C, B)
? C E A ? → **violates adjacent**(B, D)
? ? C E A → **observe** *goal*

Both this shorter version of the proof and the longer one depicted in Fig. 6 are machine-readable and can be promptly evaluated by our implementation.

9. Map coloring

As another example consider map coloring, where adjacent regions in a map must receive different colors. We will refer to this as the "adjacency constraint," or AC for short. AC can be symbolically formulated as follows:

$$\forall r_1, r_2\ .\ \mathit{adjacent}(r_1, r_2) \Rightarrow \neg \mathit{sameColor}(r_1, r_2),$$

where r_1 and r_2 range over regions and *adjacent* and *sameColor* have the obvious interpretations.[32] It is well known that any planar map can be colored in a way that satisfies AC using no more than four colors [3].[33] We will suppose that the

[30] This could be explicitly shown by appending the following sentence to the *goal* conjunction:

$$\big(\textbf{atEnd}(B) \vee \textbf{atEnd}(D) \vee \textbf{atEnd}(A)\big).$$

The deduction of Fig. 6 can successfully derive this stronger *goal* without any modification.

[31] Hence, at least one branch of the case analysis $(\sigma_j; \rho_j) \to \mathfrak{D}_j$ must have a regular deduction $\mathfrak{D}_j$ as its body, i.e., not a **violates** clause.

[32] We assume for simplicity that *adjacent* is irreflexive.

[33] Provided we do not admit disconnected regions, and that adjacency means sharing a line segment (so that, for instance, sharing a single point on the plane is not sufficient to make two regions adjacent).

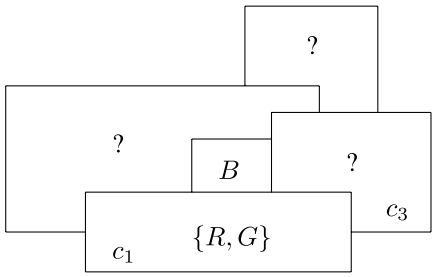


Fig. 7. An initial diagram of a map to be colored.

four available colors are red, green, blue, and yellow (R, G, B, Y). This section will illustrate how rigorous reasoning about such problems can be carried out most naturally with a mixture of diagrammatic and symbolic reasoning in Vivid[34]; it will also demonstrate how the assignment of names can be part of a heterogeneous deduction.

Consider the map of the five regions shown in Fig. 7. A questionmark inside a region indicates that the color of that region is unknown; it could be any of the four possibilities. A set of colors such as $\{R, G\}$ inside a region indicates that the color of that region must be one of the set's elements. A single color inside a region has the obvious interpretation; e.g., the central region in Fig. 7 must be colored blue. A region could optionally be given a name. In the example of Fig. 7, only two regions have names, c_1 and c_3. Suppose now that we are given the following two sentential constraints, in addition to the information depicted in the figure:

$$\neg \mathit{yellow}(c_3); \tag{36}$$

$$\mathit{sameColor}(c_1, c_2) \wedge c_1 \neq c_2. \tag{37}$$

The intended meanings are obvious: (36) requires that c_3 must not be colored yellow (hence its color must be either R, or G, or B); while (37) requires that c_1 and c_2 must be distinct regions assigned the same color. Note that the name c_2 does not appear in the diagram; it is part of the problem to figure out which region is denoted by that name. Note further that while (36) could be expressed diagrammatically, by writing $\{R, B, G\}$ inside c_3, constraint (37) cannot be expressed diagrammatically without considerable effort and clutter.

What can we infer from the given diagram, formulas (36) and (37), and AC? First, we can deduce that the colors of the two regions flanking the central blue region must be in the set $\{R, G, Y\}$; they cannot be blue by (AC). Further, the color of c_3 must be either red or green; it cannot be yellow on account of (36).

We can now perform a two-way case analysis:

1. Suppose first that the color of c_1 is red. Then, by the preceding conclusions and AC, we can conclude that c_3 must be green, while the color of the leftmost region must be yellow. Accordingly, by (37), c_2 has to be the uppermost region, since all other regions have distinct colors, and it must be red.
2. By contrast, suppose that c_1 is green. Then by similar reasoning we can conclude that c_3 must be red; the leftmost region must be yellow; and that c_2 must be the uppermost region, and it must be green.

Thus we conclude that, in either case, c_2 must be the uppermost region, and it must be either red or green, while the color of the leftmost region must be yellow. The diagram representing our final conclusion and depicting all the information we have extracted is shown in Fig. 8. Note that this is a diagrammatic conclusion that could not be expressed sententially, since it involves labeling the diagram (by placing c_2 in the appropriate region).

The above reasoning can be expressed as a heterogeneous deduction, shown in Fig. 9 and Fig. 10, where τ_0 is the initial named state shown in Fig. 7. (A detailed specification of an attribute structure, vocabulary, and interpretation will be provided shortly, but the depicted proof should be sufficiently clear at this point even without those details.) Each step of the deduction depicts the propagation of one of the constraints. Formally, the Vivid proof that expresses this reasoning is as follows:

τ_1 **by thinning with** AC;
τ_2 **by thinning with** $\neg \mathit{yellow}(c_3)$;
cases:
$\tau_3 \rightarrow$ **begin**
τ_5 **by thinning with** AC;
τ_7 **by thinning with** AC;

[34] Although this particular example pertains to map coloring, similar techniques are applicable to many constraint satisfaction problems. Vivid is particularly apt for reasoning about such problems.

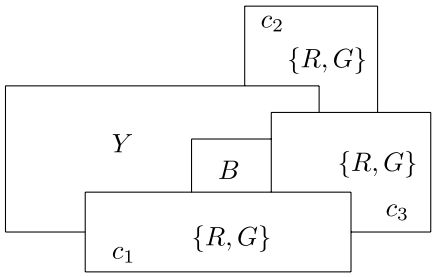


Fig. 8. A diagrammatic conclusion entailed by the diagram of Fig. 7 and constraints (36) and (37).

τ_9 **by thinning with** $sameColor(c_1, c_2) \wedge c_1 \neq c_2$;
τ_{11} **by widening**
end
$\tau_4 \rightarrow$ **begin**
τ_6 **by thinning with** AC;
τ_8 **by thinning with** AC;
τ_{10} **by thinning with** $sameColor(c_1, c_2) \wedge c_1 \neq c_2$;
τ_{11} **by widening**
end

In the interest of completeness, we close with a detailed specification of a Vivid language for map coloring. Let *Region* be a universe of regions. An appropriate attribute structure for this language is:

$$\mathcal{A}_R = \big(id : Region, neighbors : \mathcal{P}_{fin}(Region), color : \{R, B, G, Y\}; \{R_1, R_2, R_3, R_4, R_5, R_6\}\big)$$

where:

1. $R_1 \subseteq Region \times \mathcal{P}_{fin}(Region)$, defined as follows: $R_1(r, S) \Leftrightarrow r \in S$.
2. $R_2 \subseteq \{R, B, G, Y\} \times \{R, B, G, Y\}$, defined as $R_2(c_1, c_2) \Leftrightarrow c_1 = c_2$.
3. $R_3 \subseteq \{R, B, G, Y\}$ is the "red" property, i.e., $R_3(c) \Leftrightarrow c = R$. Likewise, R_4, R_5, R_6 correspond to blue, green, and yellow, respectively.

The vocabulary contains seven relation symbols: A binary *adjacent* symbol, interpreted by R_1 with profile

$$\big[(id, 1), (neighbors, 2)\big];$$

a binary *sameColor* symbol, interpreted by R_2 with profile $[(color, 1), (color, 2)]$; a binary identity symbol $=$, interpreted by the corresponding identity relation on the *id* attribute, with profile $[(id, 1), (id, 2)]$; and four unary relations, *red*, *blue*, *green*, *yellow*, interpreted by R_3–R_6, respectively, each with profile $[(color, 1)]$.

10. Related work

We have derived much inspiration from the seminal work of Barwise, Etchemendy, and others on Hyperproof [8]. Chief among the many contributions of Hyperproof were its emphasis on incomplete information and its ability to reason about ambiguous (partially determined) diagrams. These choices are not only pedagogically sound, since there are many types of problems in which students are given an incomplete sketch and are asked to fill in the gaps by way of inference; but they are also apt design choices for visual reasoning systems in general, as often the information that agents extract from a perceived image is incomplete, either because parts of the image are visually unclear or because they are not sure how to interpret them.

Important differences between Vivid and Hyperproof include the following:

1. Hyperproof is specifically built for reasoning about simple blocks worlds. Vivid, by contrast, is a domain-independent framework.
2. Hyperproof's treatment of incomplete information is ad hoc. For instance, although it is possible to signify that the size of a block is unknown, one cannot indicate that it is, say, large or medium but not small. Although these restrictions are due to the limitations of the available palette of visual abstraction conventions, they are reflected in the underlying semantics of the system [7, Section 7.5.2]. Consequently, if a new abstraction convention is added to the system, the entire semantics would need to change. By contrast, Vivid's mechanism for handling incomplete diagrammatic information via arbitrary sets of values is entirely general, and its semantics are decoupled from any particular set of abstraction conventions. Note carefully that this does not mean that an implementation of Vivid would not need to use abstraction conventions (it would), but only that, unlike Hyperproof, such conventions are not baked into the underlying semantics.

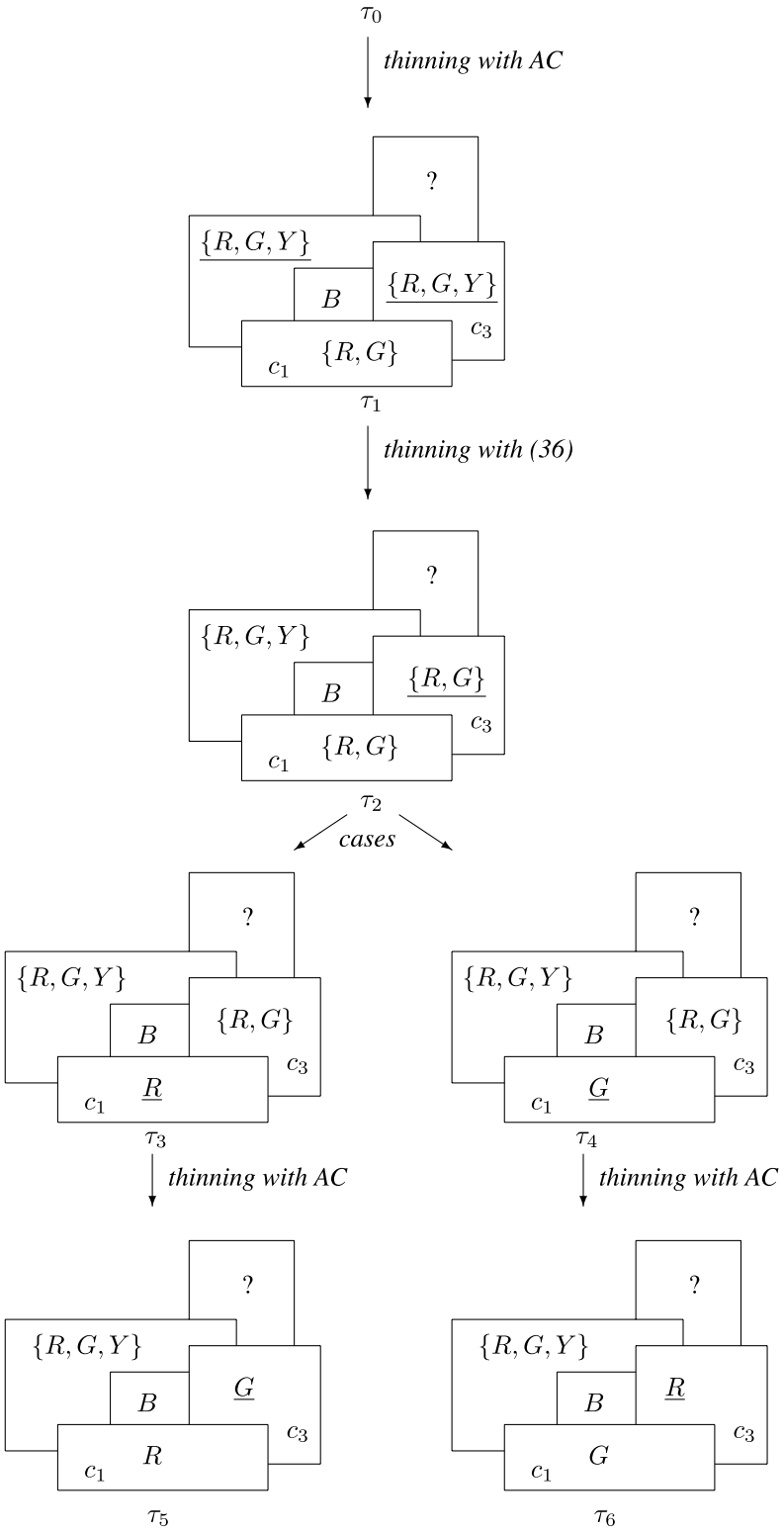


Fig. 9. Diagrammatic deduction showing the solution to a map-coloring problem, part 1.

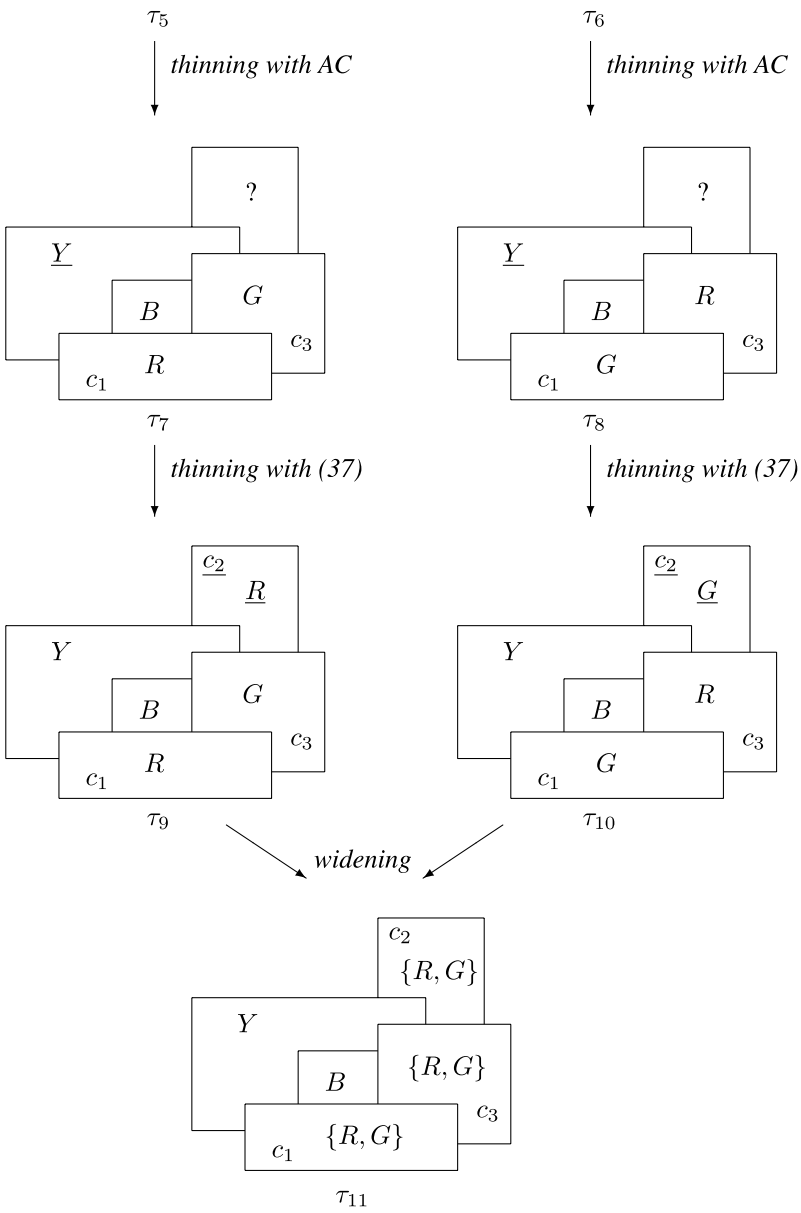


Fig. 10. Diagrammatic deduction showing the solution to a map-coloring problem, part 2.

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 reasoning [4]. More than that, a precise abstract syntax goes hand-in-hand with formal semantics, the importance of which we discuss below.
4. Vivid has a formal big-step evaluation semantics in the style of Kahn and Plotkin [33,45]. This is not to say that Hyperproof does not have precise semantics or that its semantics cannot be formally defined; only that it does not draw on the same techniques from programming language theory. We stress that this is not an issue of mere stylistic differences in presentation. Casting a formal semantics in a style such as we have used carries significant advantages, especially in meta-theoretic investigations, where many arguments take the form of induction proofs on derivations. In general, such a semantics is an invaluable tool for reasoning *about* proofs in the system, and for evaluating the correctness of algorithms that manipulate such proofs.
5. Because it is based on DPLs, Vivid is extensible from its present form as a proof-checking framework into a Turing-complete programmable system allowing the user to formulate arbitrary *methods* combining diagrammatic and sentential inference steps, in such a way that the soundness of the methods is guaranteed by the formal semantics of the language. In such a framework, heterogeneous proofs such as the solution to the seating puzzle could be discovered

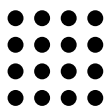
automatically. It is not clear how Hyperproof could be made programmable, let alone in a way that would guarantee soundness.

The work of Konolige and Myers on reasoning with analogical representations [40] is somewhat similar in spirit to our research, in that it seeks to formulate domain-independent principles for diagrammatic reasoning. However, they do not provide any linguistic abstractions for performing such reasoning. Rather, they outline a set of data structure operations (which they call "the integration calculus") that can be used to integrate diagrammatic inference into existing reasoning systems, and which can be described as a programming interface. By contrast, we have introduced a precisely defined family of languages for heterogeneous natural deduction, with novel syntax forms and formal semantics. Further, our method for dealing with what they call "structural uncertainty" (incomplete diagrammatic information) is much more general. Finally, our system is strictly more powerful in that it can perform diagrammatic case reasoning; their integration calculus does not have that capability.

DIAMOND [30] is a system for checking diagrammatic proofs of certain types of arithmetic theorems. The system is designed to reason about natural numbers, and specifically about universally quantified identities of the form $\forall \cdots . s = t$, 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

$$\sum_{i=1}^{n} 2i - 1 = n^2. \tag{38}$$

Diagrammatic proofs are only given for particular *instances* of the theorem, e.g., for (38) 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 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 meta-theoretic framework. This is probably the most problematic step of the process, as the problem is undecidable in general. As a result, even though DIAMOND is a proof checker and not a proof finder, it might nevertheless fail to yield a verdict. Therefore, it might make more sense to incorporate abstraction devices into the diagrams in a disciplined way, and attempt from the outset to give diagrammatic proofs of the general form of the theorem, instead of insisting on dealing with concrete diagrams only.

An attempt to extend the ideas of DIAMOND to continuous domains led to Dr. Doodle [66], which is an interactive proof assistant for metric space analysis, primarily intended as an educational tool. The inference rules are specified as "redraw rules," which can be understood as visual analogues of rewrite rules. A more extensive description of the relevant ideas is given by Winterstein, Bundy, Gurr, Jamnik [67], who also discuss an application of animation for representing and reasoning about quantification. This is an interesting idea (and novel, to the best of our knowledge), although, as the authors acknowledge, it suffers from the drawback that "it is not suited to being printed (e.g., in textbooks or papers), except as cumbersome comic strips where the simplicity of the representation is lost" [67, Section 6]. Nevertheless, we do not view this as a serious issue, insofar as a system such as Dr. Doodle is supposed to be a computerized tool. Unlike Vivid, the system is not domain-general; its inference rules are restricted to real analysis. Formal analysis of this system is provided by Winterstein [65].

GROVER [5] is a theorem-proving system that uses diagrams to guide the proof search. The system consists of a conventional (sentential) automated theorem prover (ATP), &, augmented with a diagram processor. The diagram processor examines the given diagrams and, based on the extracted information, it constructs an appropriate proof strategy for &. The system has reportedly been used to obtain automatic proofs for the diamond lemma, as well as for the Schröder–Bernstein theorem of ZF. Both are non-trivial results; the Schröder–Bernstein theorem, in particular, has a quite sophisticated sentential proof that is far from even the current state-of-the-art in ATP technology. According to the authors of GROVER, a diagram represents a trail of the objects that are involved in the proof, along with key properties of and relations among such objects. That is an interesting view of diagrams, but it differs markedly from the sense in which they are used in Vivid, where they are essentially treated as visual premises and inference rules are applied to them in the usual step-by-step fashion.

Anderson and McCartney [2] present IDR, a system for representing and computing with arbitrary diagrams. A diagram is viewed as a tessellation of a finite two-dimensional planar area, with each tile having a unique triple of numbers i, j, k

associated with it, indicating a value in the CMY (Cyan, Magenta, Yellow) color scale. Apart from the spatial relationships between the tiles, the meaning of a diagram is captured mainly via tile coloring, with different colors (or shades of gray) representing different types of information. They introduce a set of operations on diagrams, each of which takes a number of input diagrams of the same dimension and tessellation, and produces a new diagram in which the color value of a tile is a function of the color values of the corresponding tiles of the input diagrams. Among other applications, IDR has been used to solve the n-queens problem diagrammatically, to induce correct fingerings for guitar chords, and to answer queries concerning cartograms of the USA. The system is more concerned with diagram computation than with inference; there are no general notions of entailment, soundness, etc. IDR is also not heterogeneous. It is exclusively diagrammatic, in that all the available operations are applied to diagrams, not to combinations of diagrams and sentences.

Barwise and Etchemendy [6] set out to "do some of the homework needed to develop a general theory of heterogeneous inference" [6, p. 33]. They provide an information-theoretic analysis of heterogeneous inference that draws heavily from situation theory [10]. The key idea underpinning their work is a class of mathematical structures known as *infon algebras*, which turn out to be Heyting algebras (complete distributive lattices). Their analysis is thoroughly abstract, in that it is couched independently of any particular representational system and any set of syntactic constructs. Our perspective, by contrast, is that of computer science, and particularly that of artificial intelligence and programming language theory and implementation. Our chief objective is not to study heterogeneous inference from a purely mathematical perspective, but to actually design and build a formal framework that mechanizes such inference. That means coming up with an abstract syntax and with formal operational semantics, proving various results about them, and demonstrating the utility of the system via examples. Particular attention has been paid to computational efficiency concerns, implementation issues, and achieving a modular design. These issues do not surface at all in the work of Barwise and Etchemendy [6], simply because it was not the objective of that work to address them. The difference is perhaps best elucidated by the following remark of Barwise and Etchemendy [6, p. 68] on their analysis of the seating puzzle:

> We emphasize that we are presenting a mathematical model that shows the reasoning given above [solving the seating puzzle] to be valid. *This is a distinct enterprise from modeling the reasoning itself* [our italics]. It is analogous to a model-theoretic proof of the soundness of principles used in a piece of syntactic reasoning. Needless to say, this is not something that the reasoner does in the course of the reasoning.

Our focus, by contrast, has been precisely the modeling of the reasoning itself—the formal modeling of what "the reasoner does in the course of the reasoning." We want reasoners to be able to communicate certain pieces of heterogeneous reasoning to the machine succinctly and perspicuously, seamlessly integrating diagrams and sentences, in such a way that the formal machine-readable object mirrors the informal reasoning to the greatest possible extent; and we want the machine to automatically check the soundness of the reasoning. This has been our primary concern in this paper (the automatic *discovery* of such heterogeneous proofs will be our main next goal), and we believe we have achieved it, insofar, for instance, as the formal Vivid proof shown in Fig. 6 does mirror the informal reasoning quite closely, it has a perfectly rigorous syntax and semantics, and it is machine-readable and machine-checkable. The same points distinguish our work from that of Vickers [61].

Swoboda and Allwein [58] propose a general framework for modeling heterogeneous systems. The examples they present (specifically, the seating arrangement and the age charts) are readily modeled in Vivid, but a meaningful comparison is difficult, as it is not clear what proofs would look like in that framework (the paper does not present any heterogeneous deductions as examples), how soundness would be ensured (no theorems are proved about the framework), or how mechanization would be enabled.

Spider diagrams [28,55] are an extension of Euler circles and Venn diagrams, used to express constraints on sets. They arose from work on constraint diagrams [34], which were introduced as a mechanism for the visual depiction of constraints expressing invariants in object-oriented models specified in UML [47]. They combine elements of Euler circles and Venn diagrams. In particular, they relax the requirement of Venn diagrams that all curves must intersect, which retains the intuitive appeal of Euler circles, while allowing for shading, which makes Euler circles more expressive. In addition, they introduce "spiders," which are generalizations of Peirce's X-sequences (linked points, which are interpreted disjunctively as asserting that at least one of the points lies in a non-empty region). This work differs from Vivid in several respects. First, spider diagrams are a purely diagrammatic reasoning system, whereas Vivid is a heterogeneous system that integrates sentential and diagrammatic inference (a typical inference rule in Vivid, such as thinning, involves both diagrams and sentences). Second, spider diagrams, being based on Euler and Venn diagrams, are used to depict sets and relationships among sets, so their graphical elements are restricted to closed plane curves and spiders. That is, of course, quite appropriate for object-oriented design, where the main entities of interest are classes, which are naturally interpreted as sets. But there are many cases where we wish to draw and reason about other types of diagrams, say, chess boards, where the objects of interest are chess pieces rather than sets; or graphs, where the objects of interests are not sets but individual nodes and the topological relationships indicate graph-theoretic—instead of set-theoretic—relationships; or blocks worlds, where the spatial relationships of interest are on, above, and below, and potentially left-of and right-of, instead of enclosure, overlap, and separation. Such diagrams can be parsed into named system states, and once an appropriate vocabulary has been chosen, Vivid can be used to carry out heterogeneous reasoning with them. Finally, unlike spider diagrams, the semantics of Vivid are built on a 3-valued logic specifically designed to deal with incomplete information.

Wang, Lee and Zeevat [62] attempt a formal analysis of the general properties of diagrams, in the setting of multi-sorted first-order logic. They do not present a specific axiomatic characterization of diagrams, but give a few suggestive examples of such axioms, e.g., that the "overlaps" relation is symmetric, or that if a point x is inside a circle y, then x is not outside y. The authors acknowledge that, technically speaking, what they call a "graphical signature" is simply a multi-sorted signature, and what they call a "graphical theory" is simply a multi-sorted first-order theory, but they nevertheless single out some formal properties which, they argue, achieve a partial characterization of graphical theories, i.e., theories expressing diagrams. These three properties are atomicity, consistency, and maximality. Atomicity means that the only properties a diagram expresses are "basic facts," expressed by literals—atomic sentences or negations thereof. Thus, for instance, a theory that purports to capture a diagram does not contain any quantified or any conditional sentences. We agree with the general spirit of this condition, and indeed Vivid does not even use negations of atomic sentences in diagrams (system states). But there is one significant difference, namely, that Vivid allows for disjunctive information in order to accommodate incomplete diagrammatic information and visual ambiguity. Consistency simply "reflects the fact that diagrams cannot convey contradictory graphical information" [62, p. 353]. This important point also applies to Vivid diagrams, as discussed (and qualified) previously. Finally, maximality means that "for a basic property represented by an atomic sentence P about some objects in a diagram, either P or $\neg P$ can be seen in the diagram" (p. 355). That is *not* the case for Vivid diagrams, again due to the possibility of incomplete information. In Vivid, a "basic property" about "some objects in a diagram" might well have an **unknown** truth value. This also distinguishes our framework from the work of Etherington, Borgida, Brachman and Kautz [19] on "vivid knowledge bases." We believe that the suitability of a 3-valued logic for diagrams is amply demonstrated by the examples we have presented, and by systems such as Hyperproof [8]. As the underlying framework of Wang et al. [62] is that of classical first-order logic, they do not present any new inference rules, syntax forms, or semantics specifically designed for heterogeneous inference. There are no algorithms presented in the paper, and few detailed remarks on how the proposed approach would be used to mechanize diagrammatic reasoning.

Qualitative spatial reasoning [16], or QSR, is another area where ideas from heterogeneous representation and inference could suggest some interesting new research directions. QSR is an active field of research with an impressive body of work, but so far most of that work has focused on purely symbolic formalisms, and it would be interesting to explore whether the combination of diagrammatic and symbolic reasoning might have anything to contribute in this area.

11. Conclusions

Sentential reasoning has been studied to an astonishing depth over the last century, and by now we have amassed a tremendous amount of knowledge about it, both theoretical and practical. A great number of symbolic-reasoning tools such as theorem provers, proof checkers, model finders, model checkers, planners, logic programming languages for deduction, induction, and abduction, etc., are used on a daily basis to perform all kinds of tasks, from machine learning to exploring mathematics, verifying hardware and software, finding plans for autonomous agents, performing scene understanding for robots, and more.

Progress has not been as extensive for diagrammatic reasoning, even though AI researchers have long recognized the advantages of analogical representations and scientists have always used diagrams in problem solving. Indeed, as was pointed out in the introduction, visual modes of presenting information are overwhelmingly more popular (some would say trendier) than sentential ones, both in academia and in business. But in logico-mathematical domains, and particularly when it comes to formal proofs, the use of diagrams has been meager. This is partly because diagrams acquired a suspect reputation after the advent of the logicist era ushered in by Frege and Russell, which eventually resulted in an almost exclusive focus on sentential formalisms and stigmatized images as inherently non-rigorous and prone to mistakes. Attitudes towards diagrams have been improving over the last fifteen years (largely as a result of the seminal work of Barwise and Etchemendy), but the stigma has not quite disappeared. There are contemporary reports of mathematicians who go so far as "actually hiding the diagrams and visual arguments in presenting their lectures and proofs" [64, p. 281].

Our work is another step towards rectifying this discrepancy and putting diagrammatic reasoning on a solid footing by integrating it with symbolic reasoning in a heterogeneous framework. Our results demonstrate that it is possible to perform perfectly rigorous reasoning with and about an exceedingly diverse range of diagrams in combination with regular symbolic reasoning. The last dozen years or so have witnessed an encouraging surge of research activity on diagrammatic reasoning, and similar demonstrations have been carried out in certain particular diagrammatic domains such as Venn diagrams and Peirce diagrams [24,51]. Our work has broader applicability, as it introduces domain-independent idioms for heterogeneous reasoning and general notions of entailment and soundness for such reasoning.

Specifically, we have introduced and analyzed Vivid, a family of denotational proof languages (DPLs) that combine diagrammatic and symbolic inference in a mechanizable framework. Vivid is based on the notion of attribute systems, and on the use of a 3-valued logic to interpret first-order signatures into attribute structures. The design of Vivid enables highly modular implementations by enforcing a sharp separation between the purely graphical task of diagram parsing on one hand, and the system's syntax, semantics, and underlying diagrammatic inference procedures on the other. The latter are fixed once and for all and proven sound. To obtain a particular instance of Vivid, we need only specify a class of diagrams and an associated signature, interpret the signature via an appropriate attribute structure, and provide a diagram parser. This is somewhat analogous to the way in which the CLP scheme [29] defines a family of constraint logic programming

languages, with a specific member of the family obtained by fixing a constraint domain and a corresponding constraint solver and simplifier.

The combination of diagrammatic and symbolic reasoning is crucial in the process of deriving new information. A typical inference step in our system refines a diagram by using some sententially expressed knowledge to rule out certain possibilities. Incomplete information plays a key role, as reasoning in Vivid proceeds largely by the gradual elimination of uncertainty. If a diagram is completely determined, i.e., if it is a world, then it already conveys a maximal amount of information and there is not much new that can be extracted from it. That is in fact the principal limitation of Vivid at present, namely, the inability to introduce arbitrary inference rules for deriving diagrams from completely determined diagrams. This means that certain types of diagrammatic reasoning cannot presently be carried out in Vivid, e.g., various diagrammatic proofs from Euclidean geometry. Nevertheless, the capability to transform completely determined diagrams would not be problematic to incorporate into the present framework; we plan on doing just that in the near future. While such an addition would certainly extend the scope of Vivid, it may well be that some forms of diagrammatic reasoning will remain beyond its reach, or, more likely, they will not be perspicuously reproducible in it. Further work and experience with the system will be necessary before we can probe these questions more deeply.

We have not discussed how diagrams would be concretely represented within the proof text. That is an important implementation issue, but it concerns the interface of Vivid, not its abstract syntax or semantics. One possibility would be to give names to diagrams and then have those names appear in the proof text, but with hyperlinks. If a user clicks on such a link, a picture depicting the corresponding diagram would appear and the user could view or edit the diagram as necessary. Of course, as we have already stressed, how diagrams are actually drawn depends on the domain at hand and is kept separate from other aspects of the language. While this separation of concerns results in a very modular design, one might be concerned that ignoring diagram parsing amounts to brushing off potentially difficult problems that could arise in connection with diagrammatic reasoning. In response, it should be noted that diagram parsing is not quite the same as diagram reasoning, in the same way that in AI parsing formulas and reasoning with them are two different tasks. Reasoning is not concerned with how to identify the structure of a diagram or a formula, but with what to do with that structure. Just as the inference rules of sentential logic operate on the abstract syntax of formulas and reasoning starts *after* parsing has been completed, so it is with diagrams. Moreover, in very many domains diagram parsing is quite straightforward; there are no difficult issues to be addressed. Perhaps in more complicated domains diagram parsing could present challenges. But even if a framework could not handle such domains and was instead limited to working with "simple" diagrams (and Vivid has no such limitations), it could still be an exceedingly useful tool. As we pointed out in the introduction, it is not the visual complexity of diagrams that makes them useful; diagrammatic simplicity and structure are often advantages, not disadvantages. By not being fettered to any particular diagrammatic domain and any particular set of graphical conventions, Vivid gives users the freedom to exercise their creative imagination and invent their own simple diagrams and abstraction conventions, appropriate for their particular applications, and then use the machinery of the language to represent and solve problems in ways that would not be possible in a sentential framework.[35]

Two additional points should be emphasized with regard to this last remark. First, a Vivid proof in an actual computer implementation of the language would *not* contain sentential descriptions of named system states; it would contain the diagrams themselves, exactly as they appear, for instance, in a picture such as that of Fig. 7. As we mentioned above, named states would appear in the proof text essentially as hyperlinks. Clicking on such a link would launch a diagram editor, which would display the diagram and also allow the user to edit the diagram. This interface is similar to that of Hyperproof. The user sees and manipulates pictorial diagrams, *not* symbolic descriptions of system states. Of course, under the hood, Vivid will parse the diagrams into system states and will apply inference rules to those, in accordance with the theoretical framework which we have developed. Hyperproof does something similar, and indeed *any* other mechanized system would have to do something similar. But users are shielded from all that—what they see and manipulate are the diagrams themselves. Diagrams, therefore, do not serve as mere annotations to Vivid proofs, but instead play an essential role in their dynamic behavior. Second, Vivid does *not* recast diagrammatic information into a standard formalism such as multi-sorted first-order logic and then resort to the usual deductive mechanisms of that formalism for reasoning (which is what would take place in a framework such as proposed by Wang et al. [62]).[36] That approach would be unfortunate from a usability perspective; for instance, any sentential solution to the seating puzzle in first-order or higher-order logic would be much less natural than the heterogeneous solution shown in Fig. 6. Instead, Vivid represents diagrammatic information by formal structures (named system states) that are tailor-made for potentially indeterminate diagrams, and then deploys novel inference rules operating on those structures that make heterogeneous inference much more natural—all while letting the user graphically manipulate real diagrams.

[35] By drawing attention to this freedom, we should not be interpreted as assuming that the parsing of diagrams requires no ingenuity, or that such parsing can be done without regard to the meaning of diagrams. Nonetheless, diagram parsing, as opposed to the parsing of arbitrary visual scenes, is very often—though admittedly not always—a matter of fairly straightforward engineering, and the utility of the freedom Vivid provides seems to us well worth the cost of setting this engineering aside as a separate problem.

[36] For a discussion of AI and such recasting in connection with not just diagrams, but animations, see Bringsjord and Bringsjord [13].

References

[1] J. Agusti, J. Puigsegur, D.S. Robertson, A visual syntax for logic and logic programming, Journal of Visual Languages and Computing 9 (4) (1998) 399–427.
[2] M. Anderson, R. McCartney, Diagram processing: Computing with diagrams, Artificial Intelligence 145 (1–2) (2003) 181–226.
[3] K. Appel, W. Haken, Every Map is Four Colorable, first edn, American Mathematical Society, 1989.
[4] K. Arkoudas, Denotational proof languages, PhD thesis, MIT, Department of Computer Science, Cambridge, USA, 2000, available from: http://www.cag.lcs.mit.edu/~kostas/dpls/dpl-dissertation.pdf.
[5] D. Barker-Plummer, S.C. Bailin, Proofs and pictures: Proving the diamond lemma with the GROVER theorem proving system, in: Working Notes of the AAAI Spring Symposium on Reasoning with Diagrammatic Representations, American Association for Artificial Intelligence, Cambridge, MA, 1992.
[6] J. Barwise, J. Etchemendy, Information, infons, and inference, in: R. Cooper, K. Mukai, J. Perry (Eds.), Situation Theory and Its Applications, vol. 1, CSLI, Stanford, CA, 1990, pp. 33–78.
[7] J. Barwise, J. Etchemendy, Heterogeneous logic, in: J. Glasgow, N. Narayanan, N.H. Chandrasekaran (Eds.), Diagrammatic Reasoning, MIT Press, Cambridge, USA, 1995, pp. 211–234.
[8] J. Barwise, J. Etchemendy, Hyperproof: For Macintosh, CSLI Publications, 1995.
[9] J. Barwise, J. Etchemendy, Visual information and valid reasoning, in: G. Allwein, J. Barwise (Eds.), Logical Reasoning with Diagrams, Oxford University Press, 1996, pp. 3–25.
[10] J. Barwise, J. Perry, Situations and Attitudes, MIT Press, Cambridge, MA, 1983.
[11] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, Systems and Software Verification. Model-Checking Techniques and Tools, Springer, 2001.
[12] E.A. Bier, M.C. Stone, K. Pier, W. Buxton, T.D. DeRose, Toolglass and magic lenses: The see-through interface, in: Annual Conference Series, Computer Graphics 27 (1993) 73–80.
[13] S. Bringsjord, E. Bringsjord, The case against AI from imagistic expertise, Journal of Experimental and Theoretical Artificial Intelligence 8 (1996) 383–397.
[14] R.M.J. Byrne, P.N. Johnson-Laird, Spatial reasoning, Journal of Memory and Language 28 (1989) 564–575.
[15] S.-K. Chang (Ed.), Principles of Visual Programming Systems, Prentice Hall, New York, 1990.
[16] A.G. Cohn, J. Renz, Qualitative spatial representation and reasoning, in: F. van Harmelen, V. Lifschitz, B. Porter (Eds.), Foundations of Artificial Intelligence, vol. 3, Elsevier, 2008, pp. 551–596.
[17] H.G. Eggleston, Convexity, Cambridge University Press, 1969.
[18] G. Englebretsen, Linear diagrams for syllogisms (with relationals), Notre Dame Journal of Formal Logic 33 (1) (1992) 37–69.
[19] D.W. Etherington, A. Borgida, R.J. Brachman, H.A. Kautz, Vivid knowledge and tractable reasoning: Preliminary report, in: Proceedings of IJCAI-89, 10th International Joint Conference on Artificial Intelligence, Detroit, US, 1989, pp. 1146–1152.
[20] L. Euler, Lettres à une Princesse d'Allemagne, l'Academie Imperiale des Sciences, 1768.
[21] R. Fagin, A. Mendeizon, J. Ullman, A simplified universal relation assumption and its properties, ACM Transactions on Database Systems 7 (3) (1982) 343–360.
[22] A. Fish, G. Stapleton, Formal issues in languages based on closed curves, in: Proceedings of the International Workshop on Visual Languages and Computing, 2006.
[23] M. Grigni, D. Papadias, C.H. Papadimitriou, Topological inference, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, 1995, pp. 901–905.
[24] E.M. Hammer, Logic and Visual Information, CSLI Publications, Stanford, CA, 1995.
[25] D. Harel, On visual formalisms, Commun. ACM 31 (5) (1988) 514–530.
[26] P.J. Hayes, Some problems and non-problems in representation theory, in: R.J. Brachman, H.J. Levesque (Eds.), Readings in Knowledge Representation, Kaufmann, Los Altos, CA, 1985, pp. 3–22.
[27] M. Hirakawa, M. Tanaka, T. Ichikawa, An iconic programming system, HI–VISUAL, IEEE Transactions on Software Engineering 16 (10) (1990) 1178–1184.
[28] J. Howse, F. Molina, J. Taylor, S. Kent, J. Gil, Spider diagrams: A diagrammatic reasoning system, Journal of Visual Languages and Computing 12 (3) (2001) 299–324.
[29] J. Jaffar, J.-L. Lassez, Constraint logic programming, in: POPL '87: 14th ACM Symposium on Principles of Programming Languages, ACM Press, Germany, Munich, 1987, pp. 111–119.
[30] M. Jamnik, Mathematical Reasoning with Diagrams, CSLI Publications, Stanford, CA, 2001.
[31] P. Johnson-Laird, Mental Models, Harvard University Press, 1983.
[32] S.D. Johnson, J. Barwise, G. Allwein, Towards the rigorous use of diagrams in reasoning about hardware, in: G. Allwein, J. Barwise (Eds.), Logical Reasoning with Diagrams, Oxford University Press, 1996, pp. 201–223.
[33] G. Kahn, Natural semantics, in: Proceedings of Theoretical Aspects of Computer Science, Passau, Germany, 1987.
[34] S. Kent, Constraint Diagrams: Visualizing invariants in object-oriented models, in: Proceedings, Object-Oriented Programming Systems, Languages and Applications (OOPSLA '97), 1997, pp. 327–341.
[35] O. Lemon, Comparing the efficacy of visual languages, in: D. Barker-Plummer, D.I. Beaver, J. van Benthem, P.S. di Luzio (Eds.), Words, Proofs, and Diagrams, CSLI Publications, Stanford, CA, 2002, pp. 47–69.
[36] O. Lemon, I. Pratt, Spatial logic and the complexity of diagrammatic reasoning, Machine Graphics and Vision 6 (1) (1997) 89–109.
[37] B. Manbelbrot, The Fractal Geometry of Nature, W.H. Freeman and Company, 1982.
[38] K. Meinke, J.V. Tucker, Universal algebra, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), Handbook of Logic in Computer Science: Background – Mathematical Structures, vol. 1, Clarendon Press, Oxford, 1992, pp. 189–411.
[39] K. Mullet, D. Sano, Designing Visual Interfaces: Communication Oriented Techniques, Prentice Hall, 1994.
[40] K. Myers, K. Konolige, Reasoning with analogical representations, in: J. Glasgow, N. Narayanan, N.H. Chandrasekaran (Eds.), Diagrammatic Reasoning, MIT Press, Cambridge, USA, 1995, pp. 273–301.
[41] K.L. Myers, Hybrid reasoning using universal attachment, Artificial Intelligence 67 (1994) 329–375.
[42] T. Ogawa, J. Tanaka, CafePie: A visual programming system for CafeOBJ, in: Cafe: An Approach to Industrial Strength Algebraic Formal Methods, Elsevier Science, 2000, pp. 145–160.
[43] C. Peirce, The Collected Papers of C.S. Peirce, Harvard University Press, 1960.
[44] B.C. Pierce, Basic Category Theory for Computer Scientists, Foundations of Computing, MIT Press, Cambridge, MA, 1991.
[45] G.D. Plotkin, A structural approach to operational semantics, Research Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, 1981.
[46] J.C. Reynolds, Theories of Programming Languages, Cambridge University Press, 1998.
[47] J. Rumbaugh, I. Jacobson, G. Booch, The Unified Modeling Language Reference Manual, Addison-Wesley, 1999.
[48] J.M. Rushby, Formal methods and the certification of critical systems, Research report, Computer Science Laboratory, SRI International, Menlo Park, CA, 1993.

[49] B. Russell, Vagueness, Australasian Journal of Philosophy and Psychology 1 (1923) 84–92.
[50] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, Upper Saddle River, NJ, 2002.
[51] S.-J. Shin, The Logical Status of Diagrams, Cambridge University Press, 1995.
[52] S.-J. Shin, Heterogeneous reasoning and its logic, Bulletin of Symbolic Logic 10 (1) (2004) 86–106.
[53] S.-J. Shin, O. Lemon, Diagrams, in: E.N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, Stanford University, 2003, http://plato.stanford.edu/entries/diagrams.
[54] A. Sloman, Interactions between philosophy and AI: The role of intuition and non-logical reasoning in intelligence, in: Proceedings of the Second International Joint Conference on Artificial Intelligence, 1971.
[55] G. Stapleton, J. Howse, J. Taylor, S. Thompson, The expressiveness of spider diagrams, Journal of Logic and Computation 14 (6) (2004) 857–880.
[56] K. Stenning, Seeing Reason, Oxford University Press, 2002.
[57] K. Stenning, O. Lemon, Aligning logical and psychological perspectives on diagrammatic reasoning, in: Thinking with Diagrams, Kluwer, 2001.
[58] N. Swoboda, G. Allwein, Modeling heterogeneous systems, in: M. Hegarty, B. Meyer, N.H. Narayanan (Eds.), Diagrams 2002, in: LNAI, vol. 2317, Springer, 2002, pp. 131–145.
[59] E.R. Tufte, Envisioning Information, Graphics Press, 1990.
[60] M. Veltman, Diagrammatica: The Path to Feynman Rules, Cambridge Lecture Notes in Physics, vol. 4, Cambridge University Press, 1995.
[61] S. Vickers, Topology via Logic, Cambridge Tracts in Theoretical Computer Science, vol. 4, Cambridge University Press, 1989.
[62] D. Wang, J. Lee, H. Zeevat, Reasoning with diagrammatic representations, in: J. Glasgow, N. Narayanan, N.H. Chandrasekaran (Eds.), Diagrammatic Reasoning, MIT Press, Cambridge, USA, 1995, pp. 339–393.
[63] C. Ware, Information Visualization: Perception for Design, second edn, Morgan Kaufmann, 2004.
[64] G.H. Wheatley, Reasoning with images in mathematical activity, in: L.D. English (Ed.), Mathematical Reasoning: Analogies, Metaphors, and Images, Lawrence Erlbaum Associates, 1997, pp. 282–312.
[65] D. Winterstein, Using diagrammatic reasoning for theorem proving in a continuous domain, PhD thesis, The University of Edinburgh, Edinburgh, Scotland, 2004, available from: http://www.inf.ed.ac.uk/publications/thesis/online/IP040039.pdf.
[66] D. Winterstein, A. Bundy, C.A. Gurr, Dr.doodle: A diagrammatic theorem prover, IJCAR, 2004, pp. 331–335.
[67] D. Winterstein, A. Bundy, C.A. Gurr, M. Jamnik, Using animation in diagrammatic theorem proving, in: Diagrams, 2002, pp. 46–60.