Model Checking vs. Theorem Proving: A Manifesto

Article · January 2000

DOI: 10.1016/B978-0-12-450010-5.50015-3 · Source: CiteSeer

2 authors:

Joseph Y. Halpern
Cornell University
452 PUBLICATIONS **26,757** CITATIONS
SEE PROFILE

Moshe Vardi
Rice University
735 PUBLICATIONS **34,012** CITATIONS
SEE PROFILE

Model Checking vs. Theorem Proving: A Manifesto [*]

Joseph Y. Halpern and Moshe Y. Vardi
IBM Almaden Research Center
San Jose, CA 95120-6099, USA
email: {halpern,vardi}@almaden.ibm.com

ABSTRACT: We argue that rather than representing an agent's knowledge as a collection of formulas, and then doing theorem proving to see if a given formula follows from an agent's knowledge base, it may be more useful to represent this knowledge by a semantic model, and then do model checking to see if the given formula is true in that model. We discuss how to construct a model that represents an agent's knowledge in a number of different contexts, and then consider how to approach the model-checking problem.

[*] This is an extended version of a paper that appears in *Principles of Knowledge Representation and Reasoning: Procedings of the Second International Conference*, J. A. Allen, R. Fikes, and E. Sandewall (eds.), pages 325–334, 1991.

1. Introduction

The standard approach in AI to knowledge representation, going back to [McCarthy, 1968], is to represent an agent's knowledge as a collection of formulas, which we can view as a *knowledge base*. An agent is then said to know a fact if it is *provable* from the formulas in his knowledge base. This is called in [Rosenschein, 1985] the "interpreted-symbolic-structures" approach. There are two problems in applying this approach. The first comes in the difficulty of representing agents' knowledge in terms of formulas in some appropriate language. The second lies in the difficulty of theorem proving. These problems are closely related; the need to use logic to represent agents' knowledge necessitates the use of very expressive logics, but the more expressive a logic, the harder it is to prove theorems in that logic. In this paper, we argue for a model-theoretic rather than a proof-theoretic approach to the problem. Essentially, the idea is to represent the agent's knowledge by a data structure representing some semantic model (in the spirit of the "situated-automata" approach [Rosenschein, 1985; Rosenschein and Kaelbling, 1986]), and replace theorem proving by *model checking*, that is, checking whether a given formula is true in the model.

As an example of this approach, consider the context of relational database systems. Let B be a relational database and let φ be a first-order query. The theorem-proving approach would view B as representing some formula φ_B and would evaluate the query by trying to prove or disprove $\varphi_B \Rightarrow \varphi$. Unfortunately, theorem proving for first-order logic is undecidable. The model-checking approach, on the other hand, would check whether φ holds in the database B. This can be evaluated in time polynomial in the size of the data (cf. [Vardi, 1982]).

As another example, consider an agent Alice who knows that, given her current epistemic state (i.e., the information she has obtained thus far), the world could be in any one of three possible states. In the possible-worlds approach, this situation is modeled by a Kripke structure with three possible worlds. As usual, we say that Alice knows a fact p if p is true in all three of the worlds that Alice considers possible given her epistemic state.

Since, in particular, all tautologies will be true at all three worlds Alice considers possible, it follows that Alice knows all propositional tautologies. This may seem strange. The set of propositional tautologies is well-known to be co-NP-complete. How can Alice, who after all does not seem to possess any extraordinary reasoning power, know all tautologies when she can't prove them? In the model-checking approach, there is nothing unusual about this fact. Alice knows a propositional formula φ if φ is true in the three states that Alice considers possible. This can be checked in time *linear* in the length of φ. Notice that even though Alice knows all tautologies (among other facts she knows given her epistemic state), she does not know *which* of the facts she knows are tautologies (and probably does not care!). Thus, the well-known *logical omniscience* problem [Hintikka, 1975] does not present the same difficulties in the model-checking approach as it does in the theorem-proving approach (although, as we shall see, other related difficulties do arise).

The paradigm of model checking arose explicitly in the context of finite-state program verification. (See [Clarke and Grümberg, 1987] for an overview.) Suppose we have a finite-state program P (think of P as, for example, a communications protocol), and we want to know whether it satisfies some specification ψ, which we assume can be expressed in temporal logic.[1] It is not hard to completely characterize the program P by a temporal logic formula φ_P. (Essentially, φ_P describes all the possible transitions of P in each possible global state; this is possible since P is a finite-state protocol.) One way of checking whether P satisfies the specification ψ is to check if $\varphi_P \Rightarrow \psi$ is valid [Manna and Pnueli, 1981]. Unfortunately, the validity problem for temporal logic is extremely difficult. (Technically, it is exponential-time complete [Emerson and Halpern, 1985], which most likely makes it much harder than the validity problem for propositional logic.)

Later, researchers noticed that another approach would work equally well [Clarke *et al.*, 1986a; Emerson and Clarke, 1982; Queille and Sifakis, 1982]. Rather than having P be represented by a formula, P can be represented by a Kripke structure M_P: the states in M_P represent the possible global states of P, and the edges represent the possible transitions of P. Note that the *size* (i.e., the number of states) of M_P is essentially the same as the length of φ_P (viewed as a string of symbols). Checking whether P satisfies the specification ψ now amounts to checking if ψ is true at the state in M_P that corresponds to the initial state of P. This can be done in time *linear* in the size of M_P and φ.

Of course, it could be argued that all that this argument shows is that for a special subclass of formulas (namely, those of the form $\varphi_P \Rightarrow \psi$), the validity problem is significantly simpler than it is in general. Indeed, perhaps this observation should encourage us to find other subclasses for which the validity problem is solvable in polynomial time (cf. [Emerson *et al.*, 1989]). However, we would claim that this argument misses the point. The reason that formulas of the form $\varphi_P \Rightarrow \psi$ are easy

[1] Our discussion here presumes the use of *branching* temporal logic. In the case of *linear* temporal logic, the situation is somewhat more complicated, see [Emerson and Halpern, 1985; Sistla and Clarke, 1985; Lichtenstein and Pnueli, 1985].

to deal with is because φ_P characterizes a particular state in a particular structure, in that any state where φ_P is satisfied is isomorphic to the initial state of the structure defined by P. Thus, for these formulas, the validity problem reduces to a model-checking problem, and so is tractable.

The model-theoretic approach to finite-state program verification is quite practical, and several systems based on model checking have been implemented [Clarke and Grümberg, 1987; Burch *et al.*, 1992]. Moreover, it has been extended to deal with more complicated protocols and environments (including *probabilistic* protocols and assumptions of *fairness* [Vardi and Wolper, 1986]). In some cases, the assumptions that are being dealt with (such as fairness) are not even expressible in the language being used for the specification formula. This is not a hindrance to the model-theoretic approach. We can often check whether the model satisfies assumptions that are not expressible in the language.

The core of this paper (Section 2) is devoted to a detailed description of the model-checking approach, a discussion of potential problems with the approach and how some might be dealt with by using current techniques, and a comparison of the model-checking approach and the theorem-proving approach. We also consider (in Section 3) the logical omniscience problem for the model-checking point of view, and discuss a logic appropriate for reasoning using the model-checking approach, inspired by the logic of resource-bounded knowledge presented in [Moses, 1988]. We conclude in Section 4 with a general discussion of the appropriateness

2. Using the model-theoretic approach

2.1. The Muddy Children Puzzle

Before getting into technical details, we motivate the model-theoretic approach by applying it to the well-known “muddy children” puzzle. This seems particularly appropriate for this Festschrift, since McCarthy was one of the pioneers of attempting to formalize such puzzles using epistemic logics and, indeed, the muddy children puzzle is a generalization of the wise-men puzzle considered in [McCarthy, 1978]. The version of the muddy children puzzle given here is taken from [Barwise, 1981]:

> Imagine n children playing together. The mother of these children has told them that if they get dirty there will be severe consequences. So, of course, each child wants to keep clean, but each would love to see the others get dirty. Now it happens during their play that some of the

> children, say k of them, get mud on their foreheads. Each can see the mud on others but not on his own forehead. So, of course, no one says a thing. Along comes the father, who says, "At least one of you has mud on your head," thus expressing a fact known to each of them before he spoke (if $k > 1$). The father then asks the following question, over and over: "[Does any of you know whether] you have mud on your head?" Assuming that all the children are perceptive, intelligent, truthful, and that they answer simultaneously, what will happen?
>
> There is a "proof" that the first $k-1$ times he asks the question, they will all say "no", but then the k^{th} time the dirty children will answer "yes".
>
> The "proof" is by induction on k. For $k = 1$ the result is obvious: the dirty child sees that no one else is muddy, so he must be the muddy one. Let us do $k = 2$. So there are just two dirty children, a and b. Each answers "no" the first time, because of the mud on the other. But, when b says "no," a realizes that he must be muddy, for otherwise b would have known the mud was on his head and answered "yes" the first time. Thus a answers "yes" the second time. But b goes through the same reasoning. Now suppose $k = 3$; so there are three dirty children, a, b, c. Child a argues as follows. Assume I don't have mud on my head. Then, by the $k = 2$ case, both b and c will answer "yes" the second time. When they don't, he realizes that the assumption was false, that he is muddy, and so will answer "yes" on the third question. Similarly for b and c. [The general case is similar.]

There have been many attempts to describe this type of reasoning within a logic, particularly in the context of the wise-men puzzle. Besides McCarthy's own formalization in [McCarthy, 1978], other formalizations (many carried out at McCarthy's instigation) can be found in, for example, [Attardi and Simi, 1984; Aiello *et al.*, 1988; Aiello *et al.*, 1989; Kuo, 1984; Konolige, 1984; Nait Abdallah, 1989; Stark, 1981]. The wise-men puzzle is essentially a special case of the muddy children puzzle, where there are three muddy children, and the wise men are queried sequentially rather than simultaneously. Even for this special case, formalizing the puzzle is nontrivial; and all the previous formalizations involve some mechanism (typically using higher-order logic or circumscription) for capturing the deduction capabilities of the agents within the logic. We show how the situation can be characterized model-theoretically, in a particularly elegant way.

To explain our characterization, we need some preliminaries on modal logic and the possible-worlds paradigm. We want to describe an agent's knowledge. The traditional way to do this is to say that an agent knows a fact φ if it is true in all worlds that the agent considers possible. We capture this by means of a *Kripke structure*. A Kripke structure M for n agents is a tuple $(S, \pi, \mathcal{K}_1, \dots, \mathcal{K}_n)$, where S is a set of possible *worlds* or *states*, π associates with each world in S a truth assignment (propositional or first-order as the case may be), and $\mathcal{K}_i$ is a *binary relation* on S. Intuitively, the truth assignment $\pi(w)$ tells us what is true in a world w. The relation $\mathcal{K}_i$ is intended to capture the possibility relation according to agent i: $(u, v) \in \mathcal{K}_i$ if agent i considers world v possible when in world u. We then say agent i knows φ at world u in structure M, and write $(M, u) \models K_i\varphi$, if φ is true in all the worlds v that agent i considers possible in world u. This captures the intuition that an agent knows a fact if it is true at all the worlds he considers possible.

Now we return to the muddy children puzzle; much of our discussion is taken from a forthcoming book [Fagin *et al.*, 1995]. First consider the situation before the father speaks. Suppose there are n children altogether; we number them $1, \dots, n$. Some of the children have muddy foreheads, while the rest do not. We can describe a possible situation by an n-tuple of 0's and 1's of the form $(x_1, \dots, x_n)$, where $x_i = 1$ if child i has a muddy forehead, and $x_i = 0$ otherwise. Thus, if $n = 3$, then a tuple of the form $(1, 0, 1)$ would say that there are exactly two children with muddy foreheads, namely, child 1 and child 3. Suppose the actual situation is described by this tuple. What situations does child 1 consider possible before the father speaks? Since child 1 can see the foreheads of all the children besides himself, his only doubt is about whether he has mud on his own forehead. Thus child 1 considers two situations possible, namely, $(1, 0, 1)$ (the actual situation) and $(0, 0, 1)$. Similarly, child 2 considers two situations possible: $(1, 0, 1)$ and $(1, 1, 1)$. Note that in general, child i will have the same information in two possible worlds exactly if they agree in all components except possibly the i^{th} component.

We can capture the general situation by a Kripke structure M consisting of 2^n states (one for each of the 2^n n-tuples). Since child i considers a world possible if it agrees in all components except possibly the i^{th} component, we take $(s, t) \in \mathcal{K}_i$ exactly if s and t agree in all components except possibly the i^{th}-component. Notice that this definition makes $\mathcal{K}_i$ an equivalence relation. To complete the description of the Kripke structure M, we have to define the truth assignment π. To do this, we have to decide what the primitive propositions are in our language. We take them to be $\{p_1, \dots, p_n, p\}$, where, intuitively, p_i stands for "child i has a muddy forehead", while p stands for "at least one child has a muddy forehead". Thus, we define π so that $(M, (x_1, \ldots, x_n)) \models p_i$ if and only if $x_i = 1$, and $(M, (x_1, \ldots, x_n)) \models p$ if and only if $x_j = 1$ for some j. Of course, p is equivalent to $p_1 \vee \ldots \vee p_n$, so its truth value can be determined from the truth value of the other primitive propositions. There is nothing to prevent us from choosing a language where the primitive propositions are not independent. Since it is convenient to add a primitive proposition describing the father's statement, we do so. This completes the description of M.

While this Kripke structure may seem quite complicated, it actually has an elegant graphical representation. We can think of the 2^n states in S as nodes in a graph, and place an edge labeled i between states s and t if i cannot tell apart states s and t (because they agree in all components other than the i component). Suppose we ignore self-loops and the labeling on the edges for the moment. Then we have a structure with 2^n nodes, each described by an n-tuple of 0's and 1's, such that two nodes are joined by an edge exactly if they differ in one component. The reader with a good imagination will see that this defines an n-dimensional cube. The case $n = 3$ is illustrated in Figure 1.

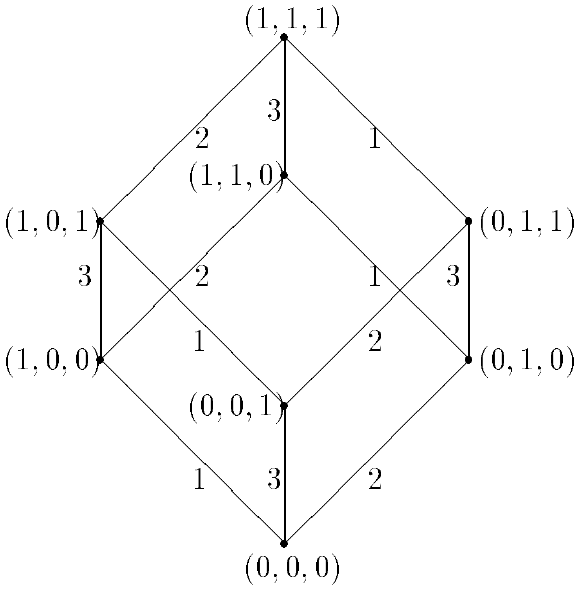

Figure 1: The Kripke structure for the muddy children puzzle with $n = 3$

Intuitively, each child knows which of the other children have muddy foreheads. This intuition is borne out in our formal definition of knowledge. For example, it is easy to see that when the actual situation is $(1, 0, 1)$, we have $(M, (1, 0, 1)) \models K_1 \neg p_2$, since in both worlds that child 1 considers possible if the actual situation is $(1, 0, 1)$, child 2 does not have a muddy forehead. Similarly, we have $(M, (1, 0, 1)) \models K_1 p_3$: child 1 knows that child 3's forehead is muddy. However, $(M, (1, 0, 1)) \models \neg K_1 p_1$. Child 1 does not know that his own forehead is muddy, since in the other world he considers possible—(0,0,1)—his forehead is not muddy.

Returning to our analysis of the puzzle, consider what happens after the father speaks. The father says p, a fact that is already known to all the children if there are two or more children with muddy foreheads. Nevertheless, the state of knowledge changes, even if all the children already know p. Going back to our example with $n = 3$, if the initial situation was $(1, 0, 1)$, although everyone knew before the father spoke that at least one child had a muddy forehead, child 1 considered the situation $(0, 0, 1)$ possible before the father spoke. In that situation, child 3 would consider $(0, 0, 0)$ possible. Thus, before the father spoke, child 1 thought it was possible that child 3 thought it was possible that none of the children had a muddy forehead. After the father speaks, it is *common knowledge* that at least one child has a muddy forehead. The notion of common knowledge, what everyone knows that everyone knows that everyone knows ..., or, in McCarthy's terminology [McCarthy *et al.*, 1979], what "any fool" knows, plays a crucial role here, as we shall see. We can represent the change in the group's state of knowledge graphically (in the general case) by simply removing the point $(0, 0, \ldots, 0)$ from the cube, getting a "truncated" cube. (More accurately, what happens is that the node $(0, 0, \ldots, 0)$ remains, but all the edges between $(0, 0, \ldots, 0)$ and nodes with exactly one 1 disappear, since it is common knowledge that even if only one child has a muddy forehead, after the father speaks that child will not consider it possible that no one has a muddy forehead.) The situation is illustrated in Figure 2.

We next show that each time the children respond to the father's question with a "no", the group's state of knowledge changes and the cube is further truncated. Consider what happens after the children respond "no" to the father's first question. We claim that now all the nodes with exactly one 1 can be eliminated. (Or, more accurately, the edges to these nodes from nodes with exactly two 1's all disappear from the graph. Nodes with one or fewer 1's are no longer reachable from nodes with two or more 1's. If there are in fact two or more children with muddy foreheads, then it is common knowledge among all the children that there are at least two children who have muddy foreheads.) The reasoning parallels that done in the "proof" given in the story. If the actual situation were described by, say, the tuple $(1, 0, \ldots, 0)$, then child 1 would initially have considered two situations possible: $(1, 0, \ldots, 0)$ and

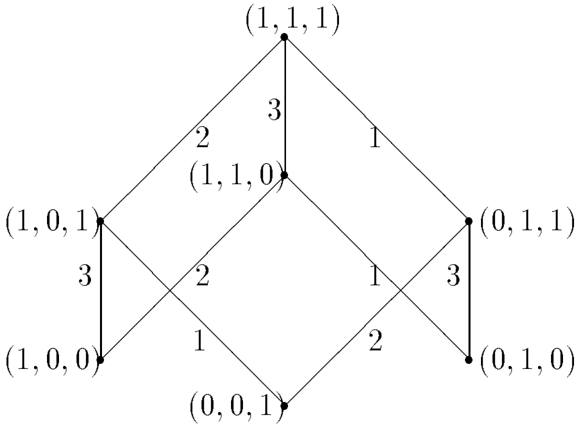

Figure 2: The Kripke structure after the father speaks

$(0, 0, \ldots, 0)$. Since once the father speaks it is common knowledge that $(0, 0, \ldots, 0)$ is not possible, he would then know that the situation is described by $(1, 0, \ldots, 0)$, and thus would know that his own forehead is muddy. Once everyone answers "no" to the father's first question, it is common knowledge that the situation cannot be $(1, 0, \ldots, 0)$. (Note that here we must use the assumption that it is common knowledge that everyone is intelligent and truthful, and so can do the reasoning required to show $(1, 0, \ldots, 0)$ is not possible.) Similar reasoning allows us to eliminate every situation with exactly one 1. Thus, after all the children have answered "no" to the father's first question, it is common knowledge that at least *two* children have muddy foreheads.

Further arguments in the same spirit can be used to show that after the children answer "no" k times, we can eliminate all the nodes with at most k 1's (or, more accurately, disconnect these nodes from the rest of the graph). We thus have a sequence of Kripke structures, describing the children's knowledge at every step in the process. Essentially, what is going on is that if, in some node s, it becomes common knowledge that a node t is impossible, then all edges from every node u reachable from s to t are eliminated. (This situation is even easier to describe once we add time to the picture. We return to this point in the next section.)

After k rounds of questioning, it is common knowledge that at least $k+1$ children have mud on their foreheads. If the true situation is described by a tuple with exactly

$k+1$ 1's, then before the father asks the question for the $(k+1)^{\mathrm{st}}$ time, those children with muddy foreheads will know the exact situation, and in particular know their foreheads are muddy, and consequently answer "yes". Note that they couldn't answer "yes" any earlier, since up to this point each child with a muddy forehead considers it possible that he did not have a muddy forehead.

We contend that this model-based argument gives a much clearer indication of what is going on with this puzzle than any of the previous formalizations that attempted to describe the theorem-proving abilities of the agents within a formal logic. The reader should note that our reasoning about the puzzle applies to any number n of children and any number $k \leq n$ of dirty children. In contrast, the theorem-proving approaches to the puzzle usually deal with some fixed k and n (typically $n = k = 3$). The ability to deal with a parametrized family of models, which is important in many instances of common-sense reasoning (cf. [Kaufmann, 1991]), is an important feature of the model-checking approach; see [Clarke *et al.*, 1986b; Sistla and German, 1987].

2.2. What is the appropriate semantic model?

The construction of the semantic model for the muddy children puzzle was somewhat *ad hoc*. Clearly, if we are to apply the model-theoretic approach in general, we need techniques for describing and constructing the semantic model, and techniques for checking if a formula is true in that model. We consider the first issue in this subsection, and leave the second to the next subsection.

Many AI applications deal with agents interacting among themselves and/or with an external environment.[2] In such a setting, we believe that the possible-worlds paradigm yields the appropriate semantic model. Thus, we would like to model a system of interacting agents as a Kripke structure. To do so, we use the formal model of [Fagin *et al.*, 1992; Halpern and Fagin, 1989] (which in turn is based on earlier models that appeared in [Halpern and Moses, 1990; Parikh and Ramanujam, 1985; Chandy and Misra, 1986; Rosenschein and Kaelbling, 1986]). We briefly describe the formal model here, since we shall make use of it later, referring the reader to [Halpern and Fagin, 1989] for more details.

We assume that at each point in time, each agent is in some *local state*. Informally,

[2] The notion of "agent" should be taken rather loosely here. An agent can be a robot observing an environment, a knowledge base (or knowledge bases) being told information, or a processor in a parallel machine. Everything we say applies in all of these contexts.

this local state encodes the information it has observed thus far. In addition, there is also an *environment* state, which keeps track of everything relevant to the system not recorded in the agents' states. A *global state* is a sequence $(s_e, s_1, \ldots, s_n)$ consisting of the environment state s_e and the local state s_i of each agent i. A *run* of the system is a function from time (which, for ease of exposition, we assume ranges over the natural numbers, although we could easily take it to range over the reals) to global states. Thus, if r is a run, then $r(0), r(1), \ldots$ is a sequence of global states that, roughly speaking, is a complete description of what happens over time in one possible execution of the system. We take a system to consist of a set of runs. Intuitively, these runs describe all the possible sequences of events that could occur in a system.

Given a system $\mathcal{R}$, we refer to a pair (r, m) consisting of a run $r \in \mathcal{R}$ and a time m as a *point*. The points can be viewed as worlds in a Kripke structure. We say two points (r, m) and (r', m') such that $r(m) = (s_e, s_1, \ldots, s_m)$ and $r'(m') = (s'_e, s'_1, \ldots, s'_m)$ are *indistinguishable* to agent i, and write $(r, m) \sim_i (r', m')$, is $s_i = s'_i$, i.e., if agent i has the same local state at both points. We take $\sim_i$ to play the role of the possibility relation $\mathcal{K}_i$; note that $\mathcal{K}_i$ is an equivalence relation under this interpretation.

An *interpreted system* is a pair $(\mathcal{R}, \pi)$ consisting of a system $\mathcal{R}$ together with a mapping π that associates a truth assignment with each point. The semantics of knowledge formulas in interpreted systems is identical to that in Kripke structures. In particular, given a point (r, m) in an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$, we have $(\mathcal{I}, r, m) \models K_i\varphi$ if $(\mathcal{I}, r', m') \models \varphi$ for all points (r', m') such that $(r', m') \sim_i (r, m)$. Notice that under this interpretation, an agent knows φ if φ is true at all the situations the system could be in, given the agent's current information (as encoded by its local state).

For a given distributed protocol, it is often relatively straightforward to construct the system corresponding to the protocol. The local state of each process can typically be characterized by a number of internal variables (that, for example, describe the messages thus far received and the values of certain local variables), and there is a transition function that describes how the system changes from one global state to another. (See, for example, [Halpern and Zuck, 1992] for a detailed example of the modeling process and a knowledge-based analysis of a distributed protocol.) In the case of the muddy children puzzle, there are 2^n runs, one corresponding to each initial situation. The n-dimensional cube we started with describes the initial situation: that is, the Kripke structure consisting just of the time 0 points. The "truncated cubes" are the Kripke structures consisting of time m points, for $m = 1, 2, 3, \ldots$ By viewing the system as a set of runs, we bring time into the picture in a natural way, and see that, in fact, nodes do not disappear, only edges. More accurately, while two runs may be joined by an edge at time m, there may not be an edge between them at time $m+1$, because the agent that couldn't distinguish them at time m acquired some information that allowed him to distinguish them at time $m+1$. (See [Halpern and Fagin, 1989] for a detailed description of the set of runs corresponding to the muddy children puzzle.)

For knowledge-based applications, the modeling problem may be harder because of the difficulty in describing the state space. Basically, it is not clear how to describe the states when they contain information about agents' knowledge. In this context, the *knowledge structures* approach suggested in [Fagin *et al.*, 1991] may prove useful (cf. [Hamilton and Delgrande, 1989; Lejoly and Minsoul, 1990]).

Suppose we have constructed a model of the system. Provided that we can completely characterize an agent's local state s by a formula φ_s, then we can reduce the problem of checking whether an agent in local state s knows φ to checking the validity of $\varphi_s \Rightarrow \varphi$, since agent i knows φ in local state s if φ is true at all possible worlds where its local state is s. However, being able to characterize an agent's local state by means of a formula will almost certainly require the use of a very expressive logic, for which theorem proving is quite intractable. The model-checking approach, on the other hand, does not require that local states be encodable as formulas. Thus, it does not require resorting to a logic that is more expressive than necessary to express the assertion φ.

An important issue to consider when comparing the two approaches is that of representation. The theorem-proving approach requires us to represent the agent's knowledge by a collection of formulas in some language. The model-checking approach instead represents the agent's knowledge as a local state in some structure. We have, however, complete freedom to decide on the representation of the local state and the structure. There are applications that arise frequently in AI where the logical choice for the representation is in terms of formulas. For example, the problem-solving system STRIPS represents a state by a set of first-order formulas; operations on the state are represented by adding certain formulas and deleting others [Fikes and Nilsson, 1972]. As another example, Levesque [Levesque, 1984a] studies knowledge bases (KBs) where we have $TELL$ and ASK operations. After the KB is told a sequence of facts, we should be able to represent the situation using the set

κ of facts that it has been told.[3] [4] What Kripke structure does this represent?

The first thing we have to decide is what the global states are going to be. There is only one agent in the system, namely, the KB itself, so a global state will be a pair consisting of the environment state and the KB's local state. As we said above, we can identify the KB's local state with the formulas that it has been told. What about the environment? Since we assume that there is a real world external to the KB, we can take the environment to be a complete description of the relevant features of the world. We model this by taking the environment state to be a *relational structure*.[5] Thus, a global state is a pair (A, κ), where A is a relational structure (intuitively, describing the world) and κ is a formula (intuitively, the conjunction of the facts the KB has been told).

Next we have to decide what the allowable global states are. This turns out to be quite sensitive to the expressiveness of the KB. Suppose we assume that the KB is *assertion-based*, i.e, it is told only facts in the assertion language and no facts about its knowledge. Then we can restrict attention to global states (A, κ) where the formula κ is true in A. (This captures the assumption that the KB is only told true facts about the world.) If we make no further restrictions, then, according to our definition of knowledge, the knowledge base KB *knows* a fact φ in a state κ if φ holds in all the global states of the form (A, κ). But that means that the KB knows φ in a state κ if φ holds in all the models of κ or, equivalently, if φ is a logical consequence of κ.

This example already shows how the model-checking approach can be viewed as

[3] To model the *evolution* of the knowledge base, one has to use some temporal structure where each point is represented by a set of formulas. See [Morris and Nado, 1986] and references therein for a discussion of such a model.

[4] Although it may indeed be reasonable in many cases to represent a situation by the set of facts an agent has been told, note that by doing so we are assuming (among other things) that the KB is being told facts about a static situation, so that *when* the information arrived is irrelevant. For example, if we consider propositions whose truth may change over time, it may be that at some point the KB is told φ and then later it is told $\neg\varphi$. We do not necessarily want to view the KB as being in an inconsistent state at this point. We can get around this difficulty by augmenting our representation to include time, so that "φ at time 3" would not be inconsistent with "$\neg\varphi$ at time 4". We can also imagine a more sophisticated KB that discards an earlier fact if it is inconsistent with later information. However, such an approach quickly leads to all the difficulties one encounters in general with updating beliefs (see, e.g., [Fagin *et al.*, 1983; Gärdenfors, 1988]). For example, what do we do if we have three facts that are pairwise consistent, but whose conjunction is inconsistent? Which of the three facts do we discard?

[5] A relational structure consists of a domain of individual elements, and an assignment of functions and relations to the function and relation symbols of the assertion language.

a generalization of the theorem-proving approach. But the model-checking approach gives us added flexibility. For example, consider the *closed-world assumption* [Reiter, 1984], where any fact not explicitly stored in the database is taken to be false. The need for such an assumption about negative information stems from the fact that in any complex application, the number of negative facts vastly outnumbers the number of positive ones, so that it is totally infeasible to explicitly represent the negative information in the database. We can capture this assumption model-theoretically by restricting attention to pairs (A, κ) where not only is κ true in A, but every atomic formula $P(d_1, \ldots, d_n)$ not implied by κ is false in A. While the closed-world assumption can be formalized in first-order logic, which means that we can use the theorem-proving approach to query closed-world knowledge bases [Reiter, 1984], analyzing the problem from the model-checking perspective leads to a complete characterization of the complexity of query answering, and furthermore, it enables one to deal with higher-order queries, which are not usually amenable to the theorem-proving approach [Vardi, 1986].

The closed-world assumption is only one of many we might consider making to restrict attention to only certain (A, κ). As Shoham [Shoham, 1987] observed, most forms of nonmonotonic reasoning can be viewed as attempts to restrict attention to some collection of *preferred* pairs. The focus in many of the works on nonmonotonic reasoning is on using some logical formalism to describe the set of preferred models. This is necessary if one wishes to apply the theorem-proving approach. Our contention is that theorem proving is just one way to evaluate queries and not necessarily the optimal way. From our point of view, the right question is not whether one can prove if an assertion φ follows from a knowledge base κ, using some nonmonotonic logic, but rather if φ holds in the Kripke structure represented by κ (where the nonmonotonicity is captured by restricting attention to a preferred set of possible worlds).

Even without the complication arising from the notion of preference, the situation gets more complicated if we assume that the KB is told facts that include information about its own knowledge, i.e., the specification is *knowledge-based.* As pointed out in [Levesque, 1984a], this can be quite important in practice. For example, suppose a KB is told the following facts about a small group of people: "John is a teacher", "Mary is a teacher", and "you know about all the teachers". Then we expect the KB to know that Bill is not a teacher (assuming it knows that Bill is distinct from Mary and John). Intuitively, this is because the three assertions together restrict the set of possible worlds to ones where only John and Mary are teachers.

How can we model this? That is, what pairs (A, κ) should we allow now (where

κ is the conjunction of the three formulas mentioned above)? Clearly, taking pairs (A, κ) such that κ is true in A will not work, since a relational structure A does not determine the truth of formulas involving knowledge. The answer is that when we have a knowledge-based specification, we cannot select the allowable states one at a time, even if we allow some notion of preference. Rather, it is the collection of all possible states that has to be consistent with the KB. Thus, we cannot just focus on the global states; we have to consider the whole resulting Kripke structure. Intuitively, we are trying to describe a Kripke structure consisting of a collection of pairs (A, κ) such that κ is true in the resulting Kripke structure. (Notice that the truth of κ depends on the whole Kripke structure, not just on A.)

Just as there are potentially many states consistent with a given assertion-based specification, there are potentially many Kripke structures consistent with a given knowledge-based specification. We can use some notion of preference to select a unique Kripke structure. A popular notion of preference is one that tries to circumscribe the KB's knowledge. The idea is to view the knowledge specified in the KB as *all that is known*. Thus, this preference can be viewed as a closed-world assumption at the knowledge level. This notion has received a lot of attention in the past decade [Fagin *et al.*, 1991; Halpern and Moses, 1984; Konolige, 1984; Levesque, 1981; Levesque, 1984a; Levesque, 1990; Parikh, 1991; Vardi, 1985].

So far we have considered only situations with a single agent. In such situations, Kripke structures degenerate to essentially sets of worlds. Many AI applications, however, deal with multiple interacting agents. It is when we try to give formal semantics to sentences such as "Dean doesn't know whether Nixon knows that Dean knows that Nixon knows about the Watergate break-in" that the full power of Kripke structures comes into play. In such situations, circumscribing the agents' knowledge is highly nontrivial (cf. [Fagin *et al.*, 1991; Parikh, 1991; Vardi, 1985]).

To summarize, in relatively simple settings it may not be too difficult to describe, implicitly or explicitly, the set of runs that characterize the system, and thus we can get a Kripke structure, which is the appropriate semantic model. Preference criteria and defaults may make this task more difficult, but we expect that even in this case, the task will become manageable as our understanding of these notions deepens.

2.3. Model checking

Suppose we have somehow constructed what we consider to be the appropriate semantic model. We now want to check if a given formula φ is true in a particular state of that model.

Checking whether an arbitrary formula in the extended language holds in a given world of a given Kripke structure has two components: we have to be able to check whether *assertions* (i.e., knowledge-free formulas) hold in a given state (we call this *assertion checking*), and check whether *knowledge* formulas (i.e., formulas of the form $K_i\varphi$) hold in a given state. If we can do both of these tasks, then we can handle arbitrary formulas by induction on the structure of the formula. Since checking whether a knowledge formula holds involves quantification over the possible worlds, the complexity of model checking depends on (and is typically polynomial in the product of) three quantities: the complexity of assertion checking, the size of the given Kripke structure, and the size of the formula. Thus, our ability to do model checking efficiently depends crucially on our ability to do assertion checking efficiently and our ability to deal with "large" structures.

For propositional languages, assertion checking is quite easy: it can be done in linear time. Adding modalities for knowledge does not significantly complicate things. Given a finite Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, define $||M||$ to be the sum of the number of states in S and the number of pairs in $\mathcal{K}_i$, $i = 1, \ldots, n$. Thus, $||M||$ is a measure of the size of the Kripke structure M. Let $|\varphi|$ be the length of φ, viewed as a string of symbols.

Proposition 2.1.: *There is an algorithm that checks if a propositional formula φ is satisfied in a structure M. The algorithm runs in time $O(||M|| \times |\varphi|)$.*

Proof: Let $\varphi_1, \ldots, \varphi_k$ be the subformulas of φ, listed in order of length, with ties broken arbitrarily. Thus, we have $\varphi_k = \varphi$, and if φ_i is a subformula of φ_j, then $i < j$. It is easy to check that there are at most $|\varphi|$ subformulas of φ, so we must have $k \leq |\varphi|$. An easy induction on k' shows that we can label each state s in M with φ_j or $\neg\varphi_j$, for $j = 1, \ldots, k'$, depending on whether or not φ_j is true at s, in time $O(k'||M||)$. The only nontrivial case is if φ_j is of the form $K_i\varphi_{j'}$, where $j' < j$. We label a state s with $K_i\varphi_{j'}$ iff each state t such that $(s,t) \in \mathcal{K}_i$ is labeled with $\varphi_{j'}$. Assuming inductively that each state has already been labeled with $\varphi_{j'}$ or $\neg\varphi_{j'}$, this step can clearly be carried out in time $O(||M||)$, as desired. ▮

Things get more complicated once we move to first-order languages. Even assertion checking may be difficult. If we consider structures with infinite domains, it is not always clear how to represent them, let alone do assertion checking.[6] Even if we

[6] Recent works [Kanellakis *et al.*, 1990; Kabanza *et al.*, 1990] address special cases of assertion checking for infinite structures.

restrict attention to structures with finite domains, assertion checking may be very difficult due to the size of the domain. In general, to check the truth of a formula such as $\forall x \varphi(x)$, we may have to check the truth of $\varphi(d)$ for each domain element d.

Even if assertion checking is easy, we still have to cope with the multitude of possible worlds. For example, in the context of incomplete knowledge bases, the number of possible worlds can be exponential in the size of the knowledge base, which makes query evaluation intractable [Vardi, 1986]. We note, however, that the fact that the number of possible worlds is quite large does not automatically mean that it is hard to check all knowledge formulas. The results of [Dwork and Moses, 1990; Moses and Tuttle, 1988] demonstrate that some knowledge formulas of interest can be evaluated efficiently in certain contexts despite an exponential number of possible worlds.

What can we do to deal with situations where the number of possible worlds or the number of domain elements is too large to handle? There are a number of approaches one could pursue. For one thing, we might hope to be able to find restricted (but still interesting) subclasses of formulas for which we can do model checking efficiently in certain restricted (but still interesting) subclasses of structures, just as we now have techniques for doing inference efficiently with certain subclasses of formulas (such as Horn clauses). Indeed, IS-A formulas (which essentially correspond to Horn clauses, although see [Brachman, 1985] for some caveats) can be checked very efficiently in IS-A hierarchies.

A second approach is to consider heuristics and defaults. Since the difficulty in model checking arises when we have large structures and/or large domains, we need good heuristics for handling such situations. We in fact use such heuristics all the time in our daily life. For example, if there are too many possibilities (i.e., too many possible worlds), we focus attention on only a few (the ones we deem to be the "most relevant" according to some metric or "most likely" according to some probability distribution), ignoring the rest. Consider a situation where Alice is asked if (she knows that) φ is the case. To answer this question, Alice would have to check all the worlds she considers possible to see if φ holds in all of them. There may be $1,000,000$ worlds that Alice considers possible, which means that she has a lot of checking to do. However, if Alice has a probability space on these worlds, then there may be a subset of worlds of size $1,000$ that has probability .99. Alice could check these $1,000$ worlds; if φ holds in all of them, then this might be taken as sufficient "evidence" for Alice to say that φ indeed holds. Even without explicit probability, Alice might still have a notion of which of the $1,000,000$ worlds are most likely or most relevant. Similarly, if we have a large domain and want to check the truth of a formula such as $\forall x\varphi(x)$, we can use heuristics and defaults to cut down the search space (perhaps checking $\varphi(d)$ for only a few potentially "abnormal" domain elements).

Yet another approach would be to have a notion of what features are "relevant" to the problem, and identify all worlds that agree on irrelevant features, thus cutting down on the search space. Being able to do this depends both on being able to identify relevant notions and on being able to check relevant features rapidly. Going back to our previous example, suppose φ is a propositional formula such as $(p_1 \vee p_2) \wedge (\neg p_2 \vee p_3)$, where p_1, p_2, and p_3 are primitive propositions. It is easy to see that φ is true unless p_1 and at least one of p_2 or p_3 are false. The truth values of all other propositions are clearly irrelevant to the truth of φ. Depending on how Alice's possible worlds are represented, it may be that Alice can check that Alice does not consider a world possible where p_1 and at least one of p_2 and p_3 are false, without checking all $1,000,000$ possible worlds. We remark that formalisms for expressing and reasoning with irrelevance have been developed, both probabilistic (for example, [Pearl and Verma, 1987]) and non-probabilistic [Subramanian and Genesereth, 1987], but this is an area where much further work remains to be done.

As all these approaches show, much depends on precisely how the worlds are generated and represented. Finding appropriate representations is, of course, a major open problem.

A great deal of effort in theorem proving has been expended on finding heuristics that work well for the formulas that arise in practice. Analogously, in model checking, it would be useful to find heuristics that work well for structures that arise in practice. Although the work on model checking in this regard is still in its infancy, the early results appear quite promising [Holtzmann, 1988; Burch *et al.*, 1992]. For example, when verifying circuits, we are not dealing with arbitrary Kripke structures. The regularity in the structure suggests heuristics that seem to work well in structures with up to 10^{20} states [Burch *et al.*, 1992]!

Of course, techniques need to be developed to allow us to use these heuristics and defaults in a principled way. Much of the recent work in "vivifying" can be viewed as being in this spirit [Etherington *et al.*, 1989; Levesque, 1986]. The idea is to cut down on the number of possibilities by filling in some (hopefully) inessential details. Work on a possible-worlds framework for probability [Bacchus, 1990; Fagin and Halpern, 1991; Halpern, 1990; Nilsson, 1986] may provide insights into using probabilistic heuristics in a principled way to cut down the search space.

Notice that by using defaults in this way, we are led naturally to nonmonotonicity. However, rather than the *logic* being nonmonotonic, the model checking is nonmonotonic. We might withdraw some of our conclusions about a formula being true in a given structure if we get further information that leads us to believe that the default assumptions we used to simplify the model-checking problem are not correct.

3. Modeling resource-bounded agents

As is well known, the standard possible-world model for knowledge suffers from the logical omniscience problem: agents know all tautologies, and know all the logical consequences of their knowledge (that is, we have the inference rule "from φ infer $K_i\varphi$" and the axiom "$K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\psi$"). The logical omniscience problem has been viewed as a major shortcoming of the possible-world framework. Various attempts have been made to overcome it (see, for example, [Fagin and Halpern, 1988; Levesque, 1984b] and the references therein). The logical omniscience problem is somewhat transformed when viewed from a model-checking perspective. As we mentioned earlier, it is no longer so unreasonable that an agent knows all tautologies (although the agent will typically not know which of the many formulas it knows to be true are in fact tautologies). Nevertheless, as we observed earlier, model checking can also be intractable. It can still be the case, given our definition, that an agent can "know" a formula without being able to compute that it knows the formula. This can be viewed as the reincarnation of the logical omniscience problem in the model-checking framework.

The source of the problem is our definition of knowledge as truth in all possible worlds. This definition does not take into account the computational effort needed to evaluate truth in all possible worlds. We would like to have a formal way of capturing the knowledge of a resource-bounded agent in the model-checking framework. In [Moses, 1988], Moses presents a logic of resource-bounded knowledge, where he tries to make sense out of notions such as "an agent can compute φ in polynomial time". His approach is very much in the spirit of the model-theoretic approach advocated here, since what it means for him to be able to compute φ in polynomial time is *not* that φ can be proved (in some appropriate axiom system) in polynomial time, but rather that the truth of φ at a particular state in a structure can be computed in polynomial time.[7] We informally discuss some details of Moses' logic here and relate it to our framework; the reader is encouraged to consult [Moses, 1988] for more

[7] A related approach to the logical omniscience problem from the theorem-proving viewpoint was proposed by Konolige [Konolige, 1986]. In his approach the agent is given some initial information. What it knows is then what it can deduce from this initial information, using a particular axiom system, in a bounded number of steps.

details and motivation.

Moses uses the distributed systems model discussed in the previous section, where a system is identified with a set of runs. As we mentioned above, the definition of what it means for $K_i\varphi$ to hold at a point in a run does not take into account complexity-theoretic considerations. We want to define a notion $K_i^{\mathcal{P}}\varphi$ which intuitively amounts to "agent i knows φ and, moreover, can compute this knowledge in polynomial time."[8] Notice that whether $(\mathcal{I}, r, m) \models K_i\varphi$ depends only on $r_i(m)$, agent i's local state at the point (r, m). If we say that agent i knows how to compute φ in polynomial time, we take this to mean that *no matter what state i is in, i can compute* φ. To capture this intuition, we require that there exist an algorithm A that gets i's local state as input, and computes in polynomial time whether φ is implied by the local state. Notice that there is a slight subtlety here: we have said that A must run in polynomial time, but we have not said what the time is polynomial in. We sidestep this issue by simply assuming that with each local state, there is some parameter; the computation must be polynomial in that parameter. For example, the parameter can be the number of agents in the system, a particular shared input (this is appropriate in cryptographic applications; see [Halpern *et al.*, 1988]), or the time (since this corresponds roughly to the number of pieces of information that the agent has received thus far, so the computation would be polynomial in the amount of information held by the agent). To summarize, given an interpreted system $\mathcal{I}$ and a point (r, m) in the system, we say that $(\mathcal{I}, r, m) \models K_i^{\mathcal{P}}\varphi$ if

1. $(\mathcal{I}, r, m) \models K_i\varphi$, and
2. there exists an algorithm A that takes as input s, a local state of agent i, and returns "Yes" or "No", depending on whether $K_i\varphi$ holds in all (resp. none) of the points where i has local state s; moreover, A runs in time polynomial in the parameter associated with s.

The logic of resource-bounded reasoning provides an elegant framework for analyzing resource-bounded notions of knowledge. It is shown to be useful for analyzing

[8] We remark that polynomial time knowledge can be viewed as a special case of explicit knowledge in the logic of general awareness of [Fagin and Halpern, 1988]. In the logic of general awareness, there are operators K_i, B_i, and A_i, where $K_i\varphi$, $B_i\varphi$ and $A_i\varphi$ represent, respectively, that agent i implicitly knows φ, explicitly knows φ, and is aware of φ. An agent explicitly knows φ exactly if he implicitly knows φ and is aware of φ; thus, the equivalence $B_i\varphi \equiv (K_i\varphi \wedge A_i\varphi)$ holds. In the logic of resource-bounded reasoning, polynomial-time knowledge ($K_i^{\mathcal{P}}$) plays the role of explicit knowledge, the usual notion of knowledge (K_i) plays the role of implicit knowledge, and "being able to compute in polynomial time" plays the role of awareness.

distributed protocols in [Moses, 1988], and it is extended in a number of ways in [Halpern *et al.*, 1988] to deal with cryptographic protocols. Some of these extensions suggest further modifications which make the logic more applicable to AI. We discuss these here.

First note that if $K_i^{\mathcal{P}}\varphi$ holds, then there is *some* polynomial time algorithm that will allow the agent to compute whether he knows φ at any state in the structure. However, the agent may not know what that algorithm is. More realistically, rather than having access to all polynomial time algorithms, an agent may have several algorithms that she can try to use to figure out whether she knows φ. It is easy to modify the logic to handle this situation. Let $\mathcal{B}$ be some set of algorithms. We then define $(\mathcal{I}, r, m) \models K_i^{\mathcal{B}}\varphi$ just as we did $(\mathcal{I}, r, m) \models K_i^{\mathcal{P}}\varphi$, except that in the second clause, we require that $A \in \mathcal{B}$, rather than that A is polynomial time. (Of course, if we take $\mathcal{B}$ to consist of all polynomial-time algorithms, then we recover the definition of $K_i^{\mathcal{P}}\varphi$.) Thus, if Alice tries exactly one of two algorithms to figure out if φ is true, then $\mathcal{B}$ would consist of those two algorithms, and $\mathcal{K}_{Alice}^{\mathcal{B}}\varphi$ would hold at a point exactly if Alice knows φ and she can compute this fact using one of her two algorithms. We can imagine that different agents have access to different algorithms; an agent with more expertise would have access to more and better algorithms. In this way, we can model a situation where one agent knows how to do something while another does not.

This framework can also be extended to capture *learning*. One way that an agent might learn is that it learns new algorithms. Thus, rather than having the class of algorithms $\mathcal{B}$ being fixed, we can imagine that it is a function of the state, which may change over time. Formally, we could expand the notion of a Kripke structure to include a function $\mathcal{B}$ that associates with each state s and agent i, the set of algorithms $\mathcal{B}(i, s)$ that agent i has at his disposal at state s. We can then define $X_i\varphi$—agent i explicitly knows φ—to be true at state s if $K_i^{\mathcal{B}(i,s)}\varphi$ holds at state s, that is, if agent i knows φ and can compute it by using one of the algorithms in $\mathcal{B}(i, s)$. We omit the formal details here; they are all straightforward.

Finally, we remark that in practice it may seem to be too strong a requirement for Alice to have to *know* φ. For example, it may be enough for Alice to know φ with high probability, or to let her algorithms make occasional mistakes, as long as they are not too frequent. It is not too hard to modify our definitions to get such probabilistic notions of computable knowledge; these issues are discussed in more detail in [Halpern *et al.*, 1988]. These probabilistic notions where error is allowed may give us a principled way to deal with the problem of having too many states in the model to do efficient model checking. This issue deserves further exploration.

4. Discussion and conclusions

We have argued here for a model-theoretic rather than a proof-theoretic approach to reasoning about knowledge. We do not mean to suggest that the model-theoretic approach is a panacea. There are difficulties to be overcome here, involving how to find the model and how to do model checking on large structures, just as there are difficulties in the theorem-proving approach. Our hope is that the model-theoretic perspective will suggest new heuristics and new approaches. We feel that many of the approaches that are currently being tried, including the preferred models approach to nonmonotonic reasoning and the idea of "vivifying" knowledge bases, are also best understood in the context of the model-checking framework. Thus, this perspective may allow us to unify a number of current lines of research.

The question still remains, for any particular application, whether to use the model-theoretic or proof-theoretic approach. At some level, it could be argued that this is a non-question; at a sufficiently high level, the two approaches converge: With a sufficiently rich logic, we can certainly characterize a model inside the logic, and there is nothing in the proof-theoretic approach that prevents us from doing our theorem proving by using model checking. Similarly, as we have suggested in the previous sections, the model-checking approach can capture theorem proving: we simply look at the class of all models consistent with the axioms. Assuming our logic has a complete axiomatization, if a fact is true in all of them, then it is provable.

However, this convergence obscures the fact that the two approaches have rather different philosophies and domains of applicability. The theorem-proving approach is more appropriate when we do not know what the models look like and the best way to describe them is by means of axioms. In contrast, the model-checking approach is appropriate when we do know what the models look like and we can describe them rather precisely, as in the muddy-children puzzle.

We would argue, however, that the model-checking approach has one major advantage over the theorem-proving approach, which is perhaps not immediately obvious from the way we have described them. The theorem-proving approach implicitly assumes that the language for describing the situation (i.e., the system or the model) is the same as the language for describing the properties of the system in which we are interested. For example, the situation calculus has been used for both purposes since McCarthy first introduced it in 1958 (when the first version of [McCarthy, 1968] appeared). That we may need rather powerful languages for describing a situation (typically far more powerful than those needed for describing the properties we want to prove about the system) is evident already in the muddy children and the wise-men puzzles.

The model-checking approach allows us to effectively decouple the description of the model from the description of the properties we want to prove about the model. This decoupling is useful in many cases besides those where circumscription is an issue. Consider the case of programming languages. Here we can view the program as describing the system. (There will be a particular set of runs corresponding to each program.) We may then want to prove properties of the program (such as termination, or some invariants). It seems awkward to require that the properties of the program be expressed in the same language as the model of the program, but that is essentially what would be required by a theorem-proving approach.

These examples suggest that when using the model-checking approach, it will probably be useful to have techniques for constructing more complicated models from simpler models, using various constructors. Work on such techniques has been a major focus in the study of programming languages for years. (A typical example is the work on CCS by Milner and his colleagues [Milner, 1980].) Indeed, in almost any discipline where complicated systems need to be analyzed, there are techniques for constructing models for a complicated system from simpler components. More emphasis on a model-checking approach might lead to more work along these lines. The hope would be that if we have a better understanding of where the models are coming from, we might be able to devise better techniques for doing model checking.

In summary, we do not expect the model-checking approach to supplant the theorem-proving approach. Rather, we would hope that techniques from each one can inspire advances in the other. In any case, we hope this manifesto inspires further research on model-checking techniques from the AI perspective.

Acknowledgments

The ideas in this paper have been brewing for a while. (A brief paragraph discussing some of these issues appeared in [Halpern, 1987].) Discussions with Hector Levesque and Yoram Moses helped us formulate these issues. We would like to thank Ron Fagin, Adam Grove, David, Israel, Steve Kaufmann, Gerhard Lakemeyer, and Hector Levesque for useful comments on an earlier draft of this paper.

References

[Aiello *et al.*, 1988] L. C. Aiello, D. Nardi, and M. Schaerf. Yet another solution to the three wisemen puzzle. In *Proc. 3rd Int'l. Symp. on Methodologies for Intelligent Systems*, pages 398–407, 1988.

[Aiello *et al.*, 1989] L. C. Aiello, D. Nardi, and M. Schaerf. Reasoning about knowledge and ignorance. In *Proc. Int'l. Conf. on Fifth Generation Systems*, pages 618–627, 1989.

[Attardi and Simi, 1984] G. Attardi and M. Simi. Reasoning across viewpoints. In *Proc. of ECAI 84*, pages 315–325, 1984.

[Bacchus, 1990] F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge.* MIT Press, Cambridge, Mass., 1990.

[Barwise, 1981] J. Barwise. Scenes and other situations. *Journal of Philosophy*, 78(7):369–397, 1981.

[Brachman, 1985] R. Brachman. I lied about the trees (or, defaults and definitions in knowledge representation). *The AI Magazine*, 6(3):80–93, 1985.

[Burch *et al.*, 1992] J. R. Burch, E. M. Clarke, D. L. Dill, J. Hwang, and K. L. McMillan. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–171, 1992.

[Chandy and Misra, 1986] K. M. Chandy and J. Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986.

[Clarke and Grümberg, 1987] E. M. Clarke and O. Grümberg. Research on automatic verification and finite-state concurrent systems. In J. F. Traub, B. J. Grosz, B. W. Lampson, and N. J. Nilsson, editors, *Annual Review of Computer Science, Vol. 2*, pages 269–289. Annual Reviews Inc., Palo Alto, Calif., 1987.

[Clarke *et al.*, 1986a] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 8(2):244–263, 1986. An early version appeared in *Proc. 10th ACM Symposium on Principles of Programming Languages*, 1983.

[Clarke *et al.*, 1986b] E. M. Clarke, O. Grümberg, and M. Browne. Reasoning about networks with many finite-state processes. In *Proc. 5th ACM Symp. on Principles of Distributed Computing*, pages 240–248, 1986.

[Dwork and Moses, 1990] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation*, 88(2):156–186, 1990.

[Emerson and Clarke, 1982] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[Emerson and Halpern, 1985] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985.

[Emerson *et al.*, 1989] E. A. Emerson, T. Sadler, and J. Srinivasan. Efficient temporal reasoning. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 166–178, 1989.

[Etherington *et al.*, 1989] D. Etherington, A. Borgida, R. J. Brachman, and H. Kautz. Vivid knowledge and tractable reasoning: preliminary report. In *Proc. Eleventh International Joint Conference on Artificial Intelligence (IJCAI '89)*, pages 1146–1152, 1989.

[Fagin and Halpern, 1988] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.

[Fagin and Halpern, 1991] R. Fagin and J. Y. Halpern. Uncertainty, belief, and probability. *Computational Intelligence*, 7(3):160–173, 1991.

[Fagin *et al.*, 1983] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *Proc. 2nd ACM Symp. on Principles of Database Systems*, pages 352–365, 1983.

[Fagin *et al.*, 1991] R. Fagin, J. Y. Halpern, and M. Y. Vardi. A model-theoretic analysis of knowledge. *Journal of the ACM*, 91(2):382–428, 1991. A preliminary version appeared in *Proc. 25th IEEE Symposium on Foundations of Computer Science*, 1984.

[Fagin *et al.*, 1992] R. Fagin, J. Y. Halpern, and M. Y. Vardi. What can machines know? On the properties of knowledge in distributed systems. *Journal of the ACM*, 39(2):328–376, 1992.

[Fagin *et al.*, 1995] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 1995.

[Fikes and Nilsson, 1972] R. E. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1972.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux*. MIT Press, Cambridge, Mass., 1988.

[Halpern and Fagin, 1989] J. Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3(4):159–179, 1989. A preliminary version appeared in *Proc. 4th ACM Symposium on Principles of Distributed Computing*, 1985, with the title "A formal model of knowledge, action, and communication in distributed systems: preliminary report".

[Halpern and Moses, 1984] J. Y. Halpern and Y. Moses. Towards a theory of knowledge and ignorance. In *Proc. AAAI Workshop on Non-monotonic Logic*, pages 125–143, 1984. Reprinted in K. R. Apt (Ed.), *Logics and Models of Concurrent Systems*, Springer-Verlag, Berlin/New York, pp. 459–476, 1985.

[Halpern and Moses, 1990] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. A preliminary version appeared in *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, 1984.

[Halpern and Zuck, 1992] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: knowledge-based derivations and correctness proofs for a family of protocols. *Journal of the ACM*, 39(3):449–478, 1992.

[Halpern *et al.*, 1988] J. Y. Halpern, Y. Moses, and M. R. Tuttle. A knowledge-based analysis of zero knowledge. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 132–147, 1988.

[Halpern, 1987] J. Y. Halpern. Using reasoning about knowledge to analyze distributed systems. In J. F. Traub, B. J. Grosz, B. W. Lampson, and N. J. Nilsson, editors, *Annual Review of Computer Science, Vol. 2*, pages 37–68. Annual Reviews Inc., Palo Alto, Calif., 1987.

[Halpern, 1990] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

[Hamilton and Delgrande, 1989] S. J. Hamilton and J. P. Delgrande. An investigation of modal structures as an alternative semantic basis for epistemic logics. *Computational Intelligence*, 5:82–96, 1989.

[Hintikka, 1975] J. Hintikka. Impossible possible worlds vindicated. *Journal of Philosophical Logic*, 4:475–484, 1975.

[Holtzmann, 1988] G. J. Holtzmann. An improved protocol reachability analysis technique. *Software Practice and Experience*, 18(2):137–161, 1988.

[Kabanza *et al.*, 1990] F. Kabanza, J.-M. Stevenne, and P. Wolper. Handling infinite temporal data. In *Proc. 9th ACM Symp. on Principles of Database Systems*, pages 392–403, 1990.

[Kanellakis *et al.*, 1990] F. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint query languages. In *Proc. 9th ACM Symp. on Principles of Database Systems*, pages 299–313, 1990.

[Kaufmann, 1991] S. G. Kaufmann. A formal theory of spatial reasoning. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. Second International Conference (KR '91)*, pages 347–356. Morgan Kaufmann, San Francisco, Calif., 1991.

[Konolige, 1984] K. Konolige. Circumscriptive ignorance. In *Proc. National Conference on Artificial Intelligence (AAAI '84)*, pages 202–204, 1984.

[Konolige, 1986] K. Konolige. *A Deduction Model of Belief.* Morgan Kaufmann, San Francisco, Calif., 1986.

[Kuo, 1984] V. Kuo. A formal natural deduction system about knowledge. Manuscript, Computer Science Dept., Stanford University, 1984.

[Lejoly and Minsoul, 1990] Ph. Lejoly and M. Minsoul. A subjective logic of knowledge. Manuscript, 1990.

[Levesque, 1981] H. J. Levesque. The interaction with incomplete knowledge bases: a formal treatment. In *Proc. Seventh International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 240–245, 1981.

[Levesque, 1984a] H. J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.

[Levesque, 1984b] H. J. Levesque. A logic of implicit and explicit belief. In *Proc. National Conference on Artificial Intelligence (AAAI '84)*, pages 198–202, 1984.

[Levesque, 1986] H. J. Levesque. Making believers out of computers. *Artificial Intelligence*, 30:81–108, 1986.

[Levesque, 1990] H. J. Levesque. All I know: a study in autoepistemic logic. *Artificial Intelligence*, 42(3):263–309, 1990.

[Lichtenstein and Pnueli, 1985] O. Lichtenstein and A. Pnueli. Checking the finite-state concurrent programs satisfy their linear specifications. In *Proc. 13th ACM Symp. on Principles of Programming Languages*, pages 97–107, 1985.

[Manna and Pnueli, 1981] Z. Manna and A. Pnueli. Verification of temporal programs: the temporal framework. In R. S. Boyer and J. S. Moore, editors, *The Correctness Problem in Computer Science*. Academic Press, New York, 1981.

[McCarthy *et al.*, 1979] J. McCarthy, M. Sato, T. Hayashi, and S. Igarishi. On the model theory of knowledge. Technical Report STAN-CS-78-657, Stanford University, 1979.

[McCarthy, 1968] J. McCarthy. Programs with common sense. In M. Minsky, editor, *Semantic Information Processing*, pages 403–418. MIT Press, Cambridge, Mass., 1968. Part of this article is a reprint from an an article by the same title, in *Proc. Conf. on the Mechanization of Thought Processes*, National Physical Laboratory, Teddington, England, Vol. 1, pp. 77–84, 1958.

[McCarthy, 1978] J. McCarthy. Formalization of two puzzles involving knowledge. Manuscript, Computer Science Dept., Stanford University, 1978.

[Milner, 1980] R. Milner. *A Calculus of Communicating Systems*. Lecture Notes in Computer Science, Vol. 92. Springer-Verlag, Berlin/New York, 1980.

[Morris and Nado, 1986] P. H. Morris and R. A. Nado. Representing actions with an assumption-based truth maintenance system. In *Proceedings, Fifth National Conference on Artificial Intelligence (AAAI '86)*, pages 13–17, 1986.

[Moses and Tuttle, 1988] Y. Moses and M. R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988.

[Moses, 1988] Y. Moses. Resource-bounded knowledge. In M. Y. Vardi, editor, *Proc. Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 261–276. Morgan Kaufmann, San Francisco, Calif., 1988.

[Nait Abdallah, 1989] M. A. Nait Abdallah. A logico-algebraic approach to the model theory of knowledge. *Theoretical Computer Science*, 66(2):205–232, 1989.

[Nilsson, 1986] N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[Parikh and Ramanujam, 1985] R. Parikh and R. Ramanujam. Distributed processing and the logic of knowledge. In R. Parikh, editor, *Proc. Workshop on Logics of Programs*, pages 256–268, 1985.

[Parikh, 1991] R. Parikh. Monotonic and nonmonotonic logics of knowledge. *Fundamenta Informaticae*, 15(3,4):255–274, 1991.

[Pearl and Verma, 1987] J. Pearl and T. Verma. The logic of representing dependencies by directed graphs. In *Proceedings, Sixth National Conference on Artificial Intelligence (AAAI '87)*, pages 374–379, 1987.

[Queille and Sifakis, 1982] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. 5th Int'l Symp. on Programming*, Lecture Notes in Computer Science, Vol. 137, pages 337–371. Springer-Verlag, Berlin/New York, 1982.

[Reiter, 1984] R. Reiter. Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modelling*, pages 191–233. Springer-Verlag, Berlin/New York, 1984.

[Rosenschein and Kaelbling, 1986] S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pages 83–97. Morgan Kaufmann, San Francisco, Calif., 1986.

[Rosenschein, 1985] S. J. Rosenschein. Formal theories of AI in knowledge and robotics. *New Generation Computing*, 3:345–357, 1985.

[Shoham, 1987] Y. Shoham. A semantical approach to nonmonotonic logics. In *Proc. 2nd IEEE Symp. on Logic in Computer Science*, pages 275–279, 1987. Reprinted in M. L. Ginsberg (Ed.), *Readings in Nonmonotonic Reasoning*, Morgan Kaufman, San Francisco, Calif., 1987, pp. 227–250.

[Sistla and Clarke, 1985] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[Sistla and German, 1987] A. P. Sistla and S. M. German. Reasoning with many processes. In *Proc. 2nd IEEE Symp. on Logic in Computer Science*, pages 138–152, 1987.

[Stark, 1981] W. R. Stark. A logic of knowledge. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 27:371–374, 1981.

[Subramanian and Genesereth, 1987] D. Subramanian and M. R. Genesereth. The relevance of irrelevance. In *Proc. Tenth International Joint Conference on Artificial Intelligence (IJCAI '87)*, pages 416–422, 1987.

[Vardi and Wolper, 1986] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st IEEE Symp. on Logic in Computer Science*, pages 332–344, 1986.

[Vardi, 1982] M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 137–146, 1982.

[Vardi, 1985] M. Y. Vardi. A model-theoretic analysis of monotonic knowledge. In *Proc. Ninth International Joint Conference on Artificial Intelligence (IJCAI '85)*, pages 509–512, 1985.

[Vardi, 1986] M. Y. Vardi. Querying logical databases. *Journal of Computer and System Sciences*, 33:142–160, 1986.