Why Is Modal Logic So Robustly Decidable?

Moshe Y. Vardi*
Department of Computer Science
Rice University
Houston, TX 77005-1892
E-mail: vardi@cs.rice.edu
URL: http://www.cs.rice.edu/~vardi

Abstract

In the last 20 years modal logic has been applied to numerous areas of computer science, including artificial intelligence, program verification, hardware verification, database theory, and distributed computing. There are two main computational problems associated with modal logic. The first problem is checking if a given formula is true in a given state of a given structure. This problem is known as the model-checking problem. The second problem is checking if a given formula is true in all states of all structures. This problem is known as the validity problem. Both problems are decidable. The model-checking problem can be solved in linear time, while the validity problem is PSPACE-complete. This is rather surprising when one considers the fact that modal logic, in spite of its apparent propositional syntax, is essentially a first-order logic, since the necessity and possibility modalities quantify over the set of possible worlds, and model checking and validity for first-order logic are computationally hard problems. Why, then, is modal logic so robustly decidable?

To answer this question, we have to take a close look at modal logic as a fragment of first-order logic. A careful examination reveals that propositional modal logic can in fact be viewed as a fragment of 2-variable first-order logic. It turns out that this fragment is computationally much more tractable than full first-order logic, which provides some explanation for the tractability of modal logic. Upon a deeper examination, however, we discover that this explanation is not too satisfactory. The tractability of modal logic is quite and cannot be explained by the relationship to two-variable first-order logic. We argue that the robust decidability of modal logic can be explained by the so-called tree-model property, and we show how the tree-model property leads to automata-based decision procedures.

*The research reported here was conducted while the author was visiting DIMACS and Bell Laboratories as part of the DIMACS Special Year on Logic and Algorithm.

1 Introduction

Modal logic, the logic of necessity and possibility, of "must be" and "may be", was discussed by several authors in ancient times, notably by Aristotle in *De Interpretatione* and *Prior Analytics*, as well as by medieval logicians. Like most work before the modern period, it was nonsymbolic and not particularly systematic in approach. The first symbolic and systematic approach to the subject appears to be the work of Lewis, beginning in 1912 and culminating in the book *Symbolic Logic* with Langford [LL59]. Propositional modal logic is obtained from propositional logic by adding a modal connective $\Box$, i.e., if ϕ is a formula, then $\Box\phi$ is also a formula. Intuitively, $\Box\phi$ asserts that ϕ is *necessarily* true. Dually, $\neg\Box\neg\Phi$, abbreviated as $\Diamond\phi$, asserts that ϕ is *possibly* true. Modal logic has many applications, due to the fact that the notions of necessity and possibility can be given many concrete interpretations. For example, "necessarily" can mean "according to the laws of physics", or "according to my knowledge", or even "after the program terminates". In the last 20 years modal logic has been applied to numerous areas of computer science, including artificial intelligence [BLMS94, MH69], program verification [CES86, Pra76, Pnu77], hardware verification [Boc82, RS83], database theory [CCF82, Lip77], and distributed computing [BAN88, HM90].

The standard semantics for modal logic is based on the possible-worlds approach, originally proposed by Carnap [Car46, Car47]. Possible-worlds semantics was further developed independently by several researchers, including [Bay58, Hin57, Hin61, Kan57, Kri59, Mer56, Mon60, Pri62], reaching its current form with Kripke [Kri63], which explains why the mathematical structures that capture the possible-worlds approach are called *Kripke structures*. The intuitive idea behind the possible-worlds model is that besides the true state of affairs, there are a number of other possible states of affairs, or "worlds". Necessity then means truth in all possible worlds. For example, an agent may be walking on the streets in San Francisco on a sunny day but may have no information at all about the weather in London. Thus, in all the worlds that the agent considers possible, it is sunny in San Francisco. On the other hand, since the agent has no information about the weather in London, there are worlds he considers possible in which it is sunny in London, and others in which it is raining in London. Thus, this agent knows that it is sunny in San Francisco, but he does not know whether it is sunny in London. Intuitively, if an agent considers fewer worlds possible, then he has less uncertainty, and more knowledge. If the agent acquires additional information—such as hearing from a reliable source that it is currently sunny in London—then he would no longer consider possible any of the worlds in which it is raining in London.

There are two main computational problems associated with modal logic. The first problem is checking if a given formula is true in a given state of a given Kripke structure. This problem is known as the *model-checking* problem. The second problem is checking if a given formula is true in all states of all Kripke structures. This problem is known as the *validity* problem. Both problems are decidable. The model-checking problem can be solved in linear time, while the validity problem is PSPACE-complete. This is rather surprising when one considers the fact that modal logic, in spite of its apparent propositional syntax, is essentially a first-order logic, since the necessity and possibility modalities quantify over the set of possible worlds, and model checking and validity for first-order logic are computationally hard problems. Furthermore, the undecidability of first-order logic is very robust. Only very restricted fragments of first-order logic are decidable, and these fragments are typically defined in terms of bounded quantifier alternation [DG79, Lew79]. The ability, however, to have arbitrary nesting of modalities in modal logic means that it does not correspond to a fragment of first-order logic with bounded quantifier alternation. Why, then, is modal logic so robustly decidable?

To answer this question, we have to take a close look at modal logic as a fragment of first-order logic. A careful examination reveals that propositional modal logic can in fact be viewed as a fragment of 2-variable first-order logic [Gab71, Ben91]. This fragment, denoted FO^2, is obtained by restricting the formulas to refer to only two individual variables. It turns out that this fragment is computationally much more tractable than full first-order logic, which provides some explanation for the tractability of modal logic.

Upon a deep examination, however, we discover that this explanation is not too satisfactory. The tractability of modal logic is quite robust and survives, for example, under various epistemic assumptions, which cannot be explained by the relationship to FO^2. To deepen the puzzle we consider an extension of modal logic, called *computation-tree logic*, or CTL [CE81]). This logic is also quite tractable, even though it is not even a first-order logic. We show that it can be viewed as a fragment of 2-variable fixpoint logic, denoted FP^2, but the latter does not enjoy the nice computational properties of FO^2. We conclude by showing that the decidability of CTL can be explained by the so-called *tree-model property*, which is enjoyed by CTL but not by FP^2. We show how the tree-model property leads to automata-based decision procedures.

2 Modal Logic

We wish to reason about worlds that can be described in terms of a nonempty finite set Φ of *propositional constants*, typically labeled $p, p', q, q', \ldots$. These propositional constants stand for basic facts about the world such as "it is sunny in San Francisco" or "Alice has mud on her forehead". We can now describe the set of modal formulas. We start with the propositional constants in Φ and form more complicated formulas by closing off under negation, conjunction, and the modal connective $\Box$. Thus, if φ and ψ are formulas, then so are $\neg\varphi$, $(\varphi \wedge \psi)$, and $\Box\varphi$. For the sake of readability, we omit the parentheses in formulas such as $(\phi \wedge \psi)$ whenever it does not lead to confusion. We also use standard abbreviations from propositional logic, such as $\varphi \vee \psi$ for $\neg(\neg\varphi \wedge \neg\psi)$, and $\varphi \rightarrow \psi$ for $\neg\varphi \vee \psi$. We take *true* to be an abbreviation for some fixed propositional tautology such as $p \vee \neg p$, and take *false* to be an abbreviation for $\neg$*true*. Also, we view possibility as the dual of necessity, and use $\Diamond\phi$ to abbreviate $\neg\Box\neg\phi$. Intuitively, ϕ is possible if $\neg\phi$ is not necessary. We can express quite complicated statements in a straightforward way using this language. For example, the formula $\Box\Diamond\Box p$ says that it is necessarily the case that possibly p is necessarily true.

Now that we have described the *syntax* of our language (that is, the set of well-formed formulas), we need *semantics*, that is, a formal model that we can use to determine whether a given formula is true or false. We formalize the semantics in terms of *Kripke structures*. A Kripke structure M is a tuple $(S, \pi, \mathcal{R})$, where S is a set of *states* (or *possible worlds*), $\pi : \Phi \rightarrow 2^S$ is an *interpretation* that associates with each propositional constant in Φ a set of states in S, and $\mathcal{R}$ is a binary relation on S, that is, a set of pairs of elements of S.

The interpretation $\pi(p)$ tells us at which state a propositional constant p is true; the intuition is that $\pi(p)$ is the set of states in which p holds. Thus, if p denotes the fact "it is raining in San Francisco", then $s \in \pi(p)$ captures the situation in which it is raining in San Francisco in the state s of the structure M. The binary relation $\mathcal{R}$ is intended to capture the possibility relation: $(s, t) \in \mathcal{R}$ if the state t is possible given the information in the state s. We think of $\mathcal{R}$ as a *possibility* relation, since it defines what states are considered possible in any given state.

We now define what it means for a formula to be true at a given state in a structure. Note that truth depends on the state as well as the structure. It is quite possible that a formula is true in one state and false in another. For example, in one state an agent may know it is sunny in San Francisco, while in another he may not. To capture this, we define the notion $(M, s) \models \varphi$, which can be read as "φ *is true at* (M, s)" or "φ *holds at* (M, s)" or "(M, s) *satisfies* φ". We define the $\models$ relation by induction on the structure of φ.

The interpretation π gives us the information we need to deal with the base case, where ϕ is a propositional constant:

$(M, s) \models p$ (for a propositional constant $p \in \Phi$) if $s \in \pi(p)$.

For conjunctions and negations, we follow the standard treatment from propositional logic; a conjunction $\psi \wedge \psi'$ is true exactly if both of the conjuncts ψ and ψ' are true, while a negated formula $\neg\psi$ is true exactly if ψ is not true:

$(M, s) \models \psi \wedge \psi'$ if $(M, s) \models \psi$ and $(M, s) \models \psi'$

$(M, s) \models \neg\psi$ if $(M, s) \not\models \psi$.

Finally, we have to deal with formulas of the form $\Box\psi$. Here we try to capture the intuition that ψ is necessarily true in state s of structure M exactly if ψ is true at all states that are possible in s. Formally, we have

$(M, s) \models \Box\psi$ if $(M, t) \models \psi$ for all t such that $(s, t) \in \mathcal{R}$.

Note that the semantics of $\Diamond$ follows from the semantics of $\Box$ and $\neg$:

$(M, s) \models \Diamond\psi$ if $(M, t) \models \psi$ for some t such that $(s, t) \in \mathcal{R}$.

One of the advantages of a Kripke structure is that it can be viewed as a labeled graph, that is, a set of labeled nodes connected by directed edges. The nodes are the states of S; the label of state $s \in S$ describes which propositional constants are true and false at s. The edges are the pairs in $\mathcal{R}$; an edge from s to t says that t is possible at s. These definitions are perhaps best illustrated by a simple example.

Suppose $\Phi = \{p, q\}$, so that our language has two propositional constants p and q. Further suppose that $M = (S, \pi, \mathcal{R})$, where $S = \{u, v, w\}$, p is true precisely at the states u and w, q is true precisely at the state u, and both v and w are possible precisely at u and v. This situation can be captured by the graph in Figure 1.

Note that we have $(M, v) \models \neg q$ and $(M, w) \models \neg q$, as q holds only in the state u. It follows that $(M, v) \models \Box\neg q$, since $\neg q$ holds in all states possible at v. Also, $(M, w) \models \Box\neg q$, since no state is possible at w. It follows that $(M, u) \models \Box\Box\neg q$, since $\Box\neg q$ holds in all states possible at u.

How hard it is to check if a given formula is true in a given state of a given Kripke structure? This problem is known as the *model-checking problem*. There is no general procedure for doing model checking in an infinite Kripke structure. Indeed, it is clearly not possible to represent arbitrary infinite structures effectively. On the other hand, in finite Kripke structures, model checking is relatively straightforward. Given a formula ϕ, define $|\phi|$, the *length* of ϕ, as the number of symbols in ϕ. Given a finite Kripke structure $M = (S, \pi, \mathcal{R})$, define $||M||$, the *size* of M, to be the sum of the number of states in S and the number of pairs in $\mathcal{R}$.

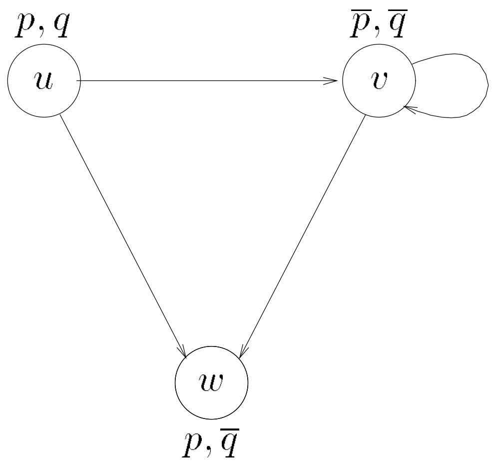

Figure 1: The Kripke structure M

Proposition 2.1 [CE81] *There is an algorithm that, given a finite Kripke structure M, a state s of M, and a modal formula ϕ, determines whether $(M, s) \models \phi$ in time $O(||M|| \times |\phi|)$.*

Proof: Let $\phi_1, \ldots, \phi_m$ be the subformulas of ϕ, listed in order of length, with ties broken arbitrarily. Thus we have $\phi_m = \phi$, and if ϕ_i is a subformula of ϕ_j, then $i < j$. There are at most $|\phi|$ subformulas of ϕ, so we must have $m \leq |\phi|$. An easy induction on k shows that we can label each state s in M with ϕ_j or $\neg\phi_j$, 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_{k+1} is of the form $\Box\phi_j$, where $j < k + 1$. We label a state s with $\Box\phi_j$ iff each state t such that $(s, t) \in \mathcal{R}$ is labeled with ϕ_j. Assuming inductively that each state has already been labeled with ϕ_j or $\neg\phi_j$, this step can clearly be carried out in time $O(||M||)$, as desired. ∎

Thus, over finite structures model checking is quite easy algorithmically.

We say that a formula ϕ is *satisfiable in* a Kripke structure M if $(M, u) \models \phi$ for some state u of M. We say that ϕ is *satisfiable* if it is satisfiable in some Kripke structure. We say that a formula ϕ is *valid in* a Kripke structure M, denoted $M \models \phi$, if $(M, u) \models \phi$ for all states u of M. We say that ϕ is *valid* if it is valid in all Kripke structures. It is easy to see that ϕ is valid iff $\neg\phi$ is unsatisfiable. The set of valid formulas can be viewed as a characterization of the logical properties of necessity (and possibility). (At this point we are considering validity over *all* Kripke structure, both finite and infinite. We will come back to this point later.)

We describe two approaches to this characterization. The first approach is *proof-theoretic*: we show that all the properties of necessity can be formally derived from a short list of basic properties. The second approach is *algorithmic*: we study algorithms that recognize properties of necessity and we consider the computational complexity of recognizing these properties.

We start by listing some basic properties of necessity.

Theorem 2.2: *For all formulas ϕ, ψ and Kripke structures M:*

(a) if ϕ is an instance of a propositional tautology, then $M \models \phi$,

(b) *if* $M \models \phi$ *and* $M \models \phi \rightarrow \psi$ *then* $M \models \psi$,

(c) $M \models (\Box\phi \land \Box(\phi \rightarrow \psi)) \rightarrow \Box\psi$,

(d) *if* $M \models \phi$ *then* $M \models \Box\phi$.

We now show that in a precise sense the properties described in Theorem 2.2 completely characterize all properties of necessity. To do so, we have to consider the notion of *provability* in an *axiom system* AX, which consists of a collection of *axioms* and *inference rules*.

Consider the following axiom system $\mathcal{K}$, which consists of the two axioms and two inference rules given below:

A1. All tautologies of propositional calculus

A2. $(\Box\phi \land \Box(\phi \rightarrow \psi)) \rightarrow \Box\psi$ (Distribution Axiom)

R1. From ϕ and $\phi \rightarrow \psi$ infer ψ (Modus ponens)

R2. From ϕ infer $\Box\phi$ (Generalization)

Theorem 2.3: [Kri63] *$\mathcal{K}$ is a sound and complete axiom system.*

While Theorem 2.3 does offer a characterization of the set of valid formulas, it is not constructive as it gives no indication of how to tell whether a given formula is indeed provable in $\mathcal{K}$. We now present results showing that the question of whether a formula is valid is *decidable*; that is, there is an algorithm that, given as input a formula ϕ, will decide whether ϕ is valid. An algorithm that recognizes valid formulas can be viewed as another characterization of the properties of necessity, one that is complementary to the characterization in terms of a sound and complete axiom system.

Our first step is to show that if a formula is satisfiable, not only is it satisfiable in some structure, but in fact it is also satisfiable in a finite structure of bounded size. (This property is called the *bounded-model property*. It is stronger than the *finite-model property*, called *finite controllability* in [DG79], which asserts that if a formula is satisfiable then it is satisfiable in a finite structure. Note that the finite-model property for implies that ϕ is valid in *all* Kripke structures if and only if it is valid in all *finite* Kripke structures. Thus, validity and finite validity coincide. This property entails decidability of validity for modal logic even without the bounded-model property, since it implies that satisfiability is recursively enumerable, and we already know from Theorem 2.3 that validity is recursively enumerable.)

Theorem 2.4: [FL79a] *If a modal formula ϕ is satisfiable, then ϕ is satisfiable in a Kripke structure with at most $2^{|\phi|}$ states.*

From Theorem 2.4, we can get an effective (although not particularly efficient) procedure for checking if a formula ϕ is valid (i.e., whether $\neg\phi$ is not satisfiable). We simply construct all Kripke structures with $2^{|\phi|}$ states (the number of such structures is finite, albeit very large), and then check if ϕ is true at each state of each of these structures. The latter check is done using the model-checking algorithm of Proposition 2.1. If ϕ is true at each state of each of these structures, then clearly ϕ is valid.

Precisely how hard is the problem of determining validity? We now offer an answer to this question. We characterize the "inherent difficulty" of the problem in terms of computational complexity.

Theorem 2.5: [Lad77] *The validity problem for modal logic is PSPACE-complete.*

Note that the upper bound is much better than the upper bound that follows from Theorem 2.4.

3 Modal Logic vs. First-Order Logic

The modal logic discussed in Section 2 is propositional, since every state is labeled by a truth assignment to the propositional constants. (Indeed, the modal logic that we presented is often called *propositional modal logic*, to distinguish it from *first-order modal logic*, in which every state is labeled by a relational structure [Gar77, HC68].) Nevertheless, modal logic is more accurately viewed as a fragment of first-order logic. Intuitively, the states in a Kripke structure correspond to domain elements in a relational structures, and modalities are nothing but a limited form of quantifiers. We now describe this connection in more detail.

Given a set Φ of propositional constants, let the vocabulary Φ^* consist of a unary predicate q corresponding to each propositional constant q in Φ, as well as a binary predicate R. Every Kripke structure M can be viewed as a relational structure M^* over the vocabulary Φ^*. More formally, we provide a mapping from a Kripke structure $M = (S, \pi, \mathcal{R})$ to a relational structure M^* over the vocabulary Φ^*. The domain of M^* is S. For each propositional constant $q \in \Phi$, the interpretation of q in M^* is the set $\pi(q)$, and the interpretation of the binary predicate R in M^* is the binary relation $\mathcal{R}$. Essentially, a Kripke structure can be viewed as a relational structure over a vocabulary consisting of one binary predicate and several unary predicates.

We now define a translation from modal formulas into first-order formulas over the vocabulary Φ^*, so that for every modal formula ϕ there is a corresponding first-order formula ϕ^* with one free variable x (ranging over S):

- $q^* = q(x)$ for a propositional constant q
- $(\neg\phi)^* = \neg(\phi^*)$
- $(\phi \wedge \psi)^* = (\phi^* \wedge \psi^*)$
- $(\Box\phi)^* = (\forall y(R(x, y) \rightarrow \phi^*(x/y)))$, where y is a new variable not appearing in ϕ^* and $\phi^*(x/y)$ is the result of replacing all free occurrences of x in ϕ^* by y.

The translation of formulas with $\Diamond$ follows from the above clauses. For example, $(\Box\Diamond q)^*$ is

$$\forall y(R(x, y) \rightarrow \exists z(R(y, z) \wedge q(z))).$$

Note that formulas with deep nesting of modalities are translated into first-order formulas with deep quantifer alternation.

Theorem 3.1: [Ben74, Ben85]

(a) $(M, s) \models \phi$ *iff* $(M^*, V) \models \phi^*(x)$*, for each assignment* V *such that* $V(x) = s$.

(b) ϕ *is a valid modal formula iff* ϕ^* *is a valid first-order formula.*

Intuitively, the theorem says that ϕ^* is true of exactly the domain elements corresponding to states s for which $(M, s) \models \phi$, and, consequently, ϕ is valid iff ϕ^* is valid.

Theorem 3.1 presents us with a seeming paradox. If modal logic is essentially a first-order logic, why is it so well-behaved computationally? Consider for example Proposition 2.1, which says that checking the truth of a modal formula in a Kripke structure can be done in time that is linear in the size of the formula and the size of the structure. In contrast, determining whether a first-order formula holds in a relational structure is PSPACE-complete [CM77]. Furthermore, while, according to Theorem 2.5, the validity problem for modal logic is PSPACE-complete, the validity problem for first-order logic is well known to be robustly undecidable [Lew79], and decidability is typically obtained only by bounding the alternation of quantifiers [DG79]. Since, as we have observed, modal logic is a first-order fragment with unbounded quantifier alternation, why is it then so robustly decidable?

To answer this question, we have to take a close look at propositional modal logic as a fragment of first-order logic. A careful examination reveals that propositional modal logic can in fact be viewed as a fragment of *2-variable first-order logic* [Gab71, Ben91]. This fragment, denoted FO^2, is obtained by restricting the formulas to refer to only two individual variables, say, x and y. Thus, $\forall x \forall y(R(x,y) \rightarrow R(y,x))$ is in FO^2, while $\forall x \forall y \forall z(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$ is not in FO^2,

To see that two variables suffice to express modal logic formulas, consider the above translation from modal logic to first-order logic. New variables are introduced there only in the fourth clause:

- $(\Box\phi)^* = (\forall y(R(x,y) \rightarrow \phi^*(x/y)))$, where y is a new variable not appearing in ϕ^* and $\phi^*(x/y)$ is the result of replacing all free occurrences of x in ϕ^* by y.

Thus, each modal connective results in the introduction of a new individual variable. For example $\Box\Box q^*$ is

$$\forall y(R(x,y) \rightarrow \forall z(R(y,z) \rightarrow q(z))),$$

which is not in FO^2. It turns out that by re-using variables we can avoid introducing new variables. All we have to do is replace the definition of ϕ^* by the definition of ϕ^+:

- $q^+ = q(x)$ for a propositional constant q
- $(\neg\phi)^+ = \neg(\phi^+)$
- $(\phi \wedge \psi)^+ = (\phi^+ \wedge \psi^+)$
- $(\Box\phi)^+ = (\forall y(R(x,y) \rightarrow \forall x(x = y \rightarrow \phi^+)))$.

Thus, $(\Box\Box q)^+$ is

$$\forall y(R(x,y) \rightarrow \forall x(x = y \rightarrow \forall y(R(x,y) \rightarrow \forall x(x = y \rightarrow q(x))).$$

Theorem 3.2: [Gab71]

(a) $(M,s) \models \phi$ *iff* $(M^*, V) \models \phi^+(x)$*, for each assignment* V *such that* $V(x) = s$.

(b) ϕ *is a valid modal formula iff* ϕ^+ *is a valid* FO^2*-formula.*

How does the fact that modal logic can be viewed as a fragment of FO^2 explain its computational properties? Consider first the complexity of evaluating truth of formulas. By Proposition 2.1, truth modal formulas can be evaluated in time that is linear in the size of the structure and in the size of the formula. How hard it is to evaluate truth of FO^2 formulas?

Proposition 3.3 [Imm82, Var95] *There is an algorithm that, given a relational structure M over a domain D, an FO^2-formula $\phi(x,y)$, and an assignment $V : \{x,y\} \rightarrow D$, determines whether $(M,V) \models \phi$ in time $O(||M||^2 \times |\phi|)$.*

Thus, truth of FO^2-formulas can be evaluated efficiently (in contrast to the truth general first-order formulas, whose evaluation, as we said, is PSPACE-complete), though not as efficiently as the truth of modal formulas. It is an intriguing question whether there is an interesting fragment of FO^2 whose model-checking problem can be solved in linear time.

Does the embedding in FO^2 explains the decidability of modal logic? The first decidability result for FO^2 was obtained by Scott [Sco62], who showed that the decision problem for FO^2 can be reduced to that of the Gödel class, i.e., the class of first-order sentences with quantifier prefix of the form $\exists^*\forall\forall\exists^*$. Since, the Gödel class without equality is decidable [Göd32, Kal33, Sch34], Scott's reduction yields the decidability of FO^2 without equality. This proof does not extend to FO^2 with equality, since the Gödel class with equality is undecidable [Gol84]. The full class FO^2 with equality was considered by Mortimer [Mor75]. He proved that this class is decidable by showing that it has the finite-model property: if an FO^2-formula (with equality) is satisfiable, then it is satisfiable by a finite model. Thus, validity and finite validity coincide for FO^2. An analysis of Mortimer's proof shows that he actually established a bounded-model property property for FO^2: if an FO^2-formula with equality is satisfiable, then it is satisfiable by a model whose size is at most doubly exponential in the length of ϕ. More recently, this result was improved:

Theorem 3.4: [GKV96] *If an FO^2-formula ϕ is satisfiable, then ϕ is satisfiable in a relational structure with at most $2^{|\phi|}$ elements.*

Thus, to check whether an FO^2-formula ϕ is valid one has to check only all structures of exponential size. By Theorem 3.2, Theorem 3.4 implies Theorem 2.4, since the translation from modal logic to FO^2 is linear. Note, however, the validity problem for FO^2 is hard for co-NEXPTIME [Für81], and consequently, by Theorem 3.4, co-NEXPTIME complete, while the validity problem for modal logic is PSPACE-complete (Theorem 2.5). Thus, the validity problem for modal logic is probably easier than the validity problem for FO^2.

The embedding of modal logic in FO^2 does seem to offer an explanation to the computational tractability of modal logic. It turns out, however, that this explanation is of limited scope. To see why, let us recall that the versatility of modal logics stems from its adaptability to many specific applications. Consider, for example, necessity as knowledge (as in *epistemic logic* [Hin62, FHMV95]). In that case, we may want $\Box$ to have properties beyond the minimal set of properties given by the axiom system $\mathcal{K}$. For example, an important property of knowledge is *veracity*, i.e., what is known should be true. Formally, we want the property $\Box\phi \rightarrow \phi$ to hold for necessity as knowledge.

An important observation made around 1960 by modal logicians (cf. [Hin57, Hin61, Kan57, Kri63]) was that the logical properties of necessity are intimately related to the graph-theoretical properties of the possibility relations in Kripke structures. For example, veracity is related to reflexivity of the possibility relations. Let us make this claim precise.

Let $\mathcal{M}$ be a class of Kripke structures. We say that a modal formula ϕ is *valid* in $\mathcal{M}$ if it is valid in all Kripke structures in $\mathcal{M}$. An axiom system AX is said to be *sound* for $\mathcal{M}$ if every provable formula is valid in $\mathcal{M}$, and it is said to be *complete* for $\mathcal{M}$ if every formula that is valid in $\mathcal{M}$ is provable.

A Kripke structure $M = (S, \pi, \mathcal{R})$ is said to be reflexive if the possibility relation $\mathcal{R}$ is reflexive. Let $\mathcal{M}_r$ be the class of all reflexive Kripke structures. Let $\mathcal{T}$ be the axiom system obtained from $\mathcal{K}$ by adding the axiom of veracity: $\Box p \rightarrow p$.

Theorem 3.5: [Che80] *$\mathcal{T}$ is sound and complete for $\mathcal{M}_r$.*

Thus $\mathcal{T}$ characterizes all the properties of necessity in reflexive Kripke structures. Let us now examine this characterization from the complexity-theoretic perspective. How hard it is to determine validity of modal formulas under the assumption of veracity?

Theorem 3.6: [Lad77] *The validity problem for modal logic in $\mathcal{M}_r$ is PSPACE-complete.*

Does the embedding in FO^2 explain the decidability of validity in reflexive structures? Indeed it does, since reflexivity can easily be expressed by an FO^2-sentence.

Proposition 3.7 *A modal formula ϕ is valid in $\mathcal{M}_r$ iff the FO^2-formula $\forall x(R(x, x)) \rightarrow \phi^+$ is valid.*

So far, so good; FO^2 still seems to explain the decidability of modal logic. Unfortunately, this explanation breaks down when we consider other properties of necessity. Consider for example the properties of introspection, i.e., knowledge about knowledge.

- Positive introspection – "I know what I know": $\Box p \rightarrow \Box\Box p$.
- Negative introspection – "I know what I don't know": $\neg\Box p \rightarrow \Box\neg\Box p$.

A Kripke structure $M = (S, \pi, \mathcal{R})$ is said to be reflexive-symmetric-transitive if the possibility relation is reflexive, symmetric and transitive. Let $\mathcal{M}_{rst}$ be the class of all reflexive-symmetric-transitive Kripke structures. Let $\mathcal{S}5$ be the axiom system obtained from $\mathcal{T}$ by adding the two axiom of introspection: $\Box p \rightarrow \Box\Box p$ and $\neg\Box p \rightarrow \Box\neg\Box p$.

Theorem 3.8:

1. [Che80] *$\mathcal{S}5$ is sound and complete for $\mathcal{M}_{rst}$.*
2. [Lad77] *The validity problem for modal logic in $\mathcal{M}_{rst}$ is NP-complete.*

Note that while symmetry can be expressed by the FO^2-sentence $\forall x \forall y(R(x, y) \rightarrow R(y, x))$, transitivity requires the sentence $\forall x \forall y \forall z(R(x, y) \wedge R(y, z) \rightarrow R(x, z))$, which is not in FO^2. Thus, validity of modal formulas in $\mathcal{M}_{rst}$ cannot be reduced to validity of FO^2-formulas.

In general, the decidability of modal logic is very, very robust (cf. [HM92, Var89]). As a rule of thumb, the validity problem for a modal logic is typically decidable; one has to make an effort to find a modal logic with an undecidable validity problem (cf. [HV89, LR86]. The translation of modal logic to FO^2 provides a very partial explanation for this robustness (see also [ABN95] for another partial explanation in terms of *bounded quantification*, but so far no general explanation for this robust decidability is known. We deepen the puzzle in the next section.

4 Computation Tree Logic

A Kripke structure $M = (S, \pi, \mathcal{R})$ can be viewed as an operational model for a program: S is the sets of states that the program can be in, where a state is a snapshot of the relevant part of the program run-time environment, i.e., the memory, the registers, the program stack, and the like, π described which events are observable in each state, and R describes transitions that can occur by executing one step of the program. Such a model is called a *transition system* [Kel76]; it abstracts away the syntax of the program and focuses instead on its operational behavior. As such it is appropriate for modeling both software and hardware. Note that transition systems are *nondeterministic*; a node s can have more than one outgoing $\mathcal{R}$-edge. This is a powerful abstraction mechanism that let us model the uncertainty about the environment in which the program is running. For example, nondeterminism can arise from the many possible interleaving between concurrent processes. For technical convenience, we assume that the program never deadlocks by requiring that $\mathcal{R}$ be *total*, i.e., that every state has an outgoing $\mathcal{R}$-edge. This assumption does not restrict the modeling power of the formalism, since we can view a terminated execution as repeating forever its last state by adding a self-loop to that state. In numerous applications, notably integrated circuits and protocols for communication, coordination, and synchronization, the system consists of only finitely many states [Liu89, Rud87], giving rise to *finite-state systems*.

Temporal logics, which are modal logics geared towards the description of the temporal ordering of events, have been adopted as a powerful tool for specifying and verifying concurrent programs [Pnu77, MP92]. We distinguish between two types of temporal logics: linear and branching [Lam80]. In linear temporal logics, each moment in time has a unique possible future, while in branching temporal logics, each moment in time may split into several possible futures. Our focus here is on a particular branching temporal logic, called *computation tree logic*, or CTL for short [CE81].

CTL provides branching temporal connectives that are composed of a path quantifier immediately followed by a single linear temporal connective [Eme90]. The path quantifiers are A ("for all paths") and E ("for some path"). The linear-time connectives are X ("next time") and U ("until"). For example, the formula $E(pUq)$ says that there is a computation along which p holds until q holds. Formally, given a set Φ of propositional constants, a CTL-formula is one of the following:

- p, for all $p \in \Phi$,
- $\neg\psi$ or $\psi \wedge \psi'$, where ψ and ψ' are CTL-formulas.
- $EX\psi$, $AX\psi$, $E(\psi U \psi')$, $A(\psi U \psi')$, where ψ and ψ' are CTL-formulas.

To define satisfaction of CTL-formulas in a Kripke structures $M = (S, \pi, \mathcal{R})$, we need the notion of a *path* in a M. A path in M is an infinite sequence $s_0, s_1, \ldots$ of states such that for every $i \geq 0$, we have that $(s_i, s_{i+1}) \in \mathcal{R}$.

A state s in a Kripke structure $M = (S, \pi, \mathcal{R})$ satisfies a CTL-formula ϕ, denoted $(M, s) \models \phi$, under the following conditions:

- $(M, s) \models p$ (for $p \in \Phi$) if $s \in \pi(p)$.
- $(M, s) \models \psi \wedge \psi'$ iff $(M, s) \models \psi$ and $(M, s) \models \psi'$.
- $(M, s) \models \neg\psi$ if $(M, s) \not\models \psi$.

- $(M, s) \models EX\psi$ if $(M, t) \models \psi$ for some t such that $(s, t) \in \mathcal{R}$.
- $(M, s) \models AX\psi$ if $(M, t) \models \psi$ for all t such that $(s, t) \in \mathcal{R}$.
- $(M, s) \models E(\psi U \psi')$ if there exists a path $s_0, s_1, \ldots$, with $s_0 = s$, and some $i \geq 0$, such that $(M, s_i) \models \psi'$, and for all j, where $0 \leq j < i$, we have $(M, s_j) \models \psi$.
- $(M, s) \models A(\psi U \psi')$ if for all paths $s_0, s_1, \ldots$, with $s_0 = s$, there exists some $i \geq 0$, such that $(M, s_i) \models \psi'$, and for all j, where $0 \leq j < i$, we have $(M, s_j) \models \psi$.

Note that EX and AX correspond to the $\Diamond$ and $\Box$ of modal logic, while EU and AU have no such counterparts.

CTL enables us to make powerful assertions about the behavior of the program. For example, $E(\mathit{true} U \phi)$ says that along some computation ϕ eventually holds. This is abbreviated by $EF\phi$. It may seem that all the temporal connectives talk about finite computations (since X has to be satisfied in one program step and U has to be satisfied in finitely many program steps), but we can combine temporal and Boolean connectives to form assertions about infinite computations. For example, consider the formula $A(\mathit{true} U \neg\phi)$. This formula holds in a state s if along every path starting at s eventually $\neg\phi$ holds, which is abbreviated as $AF\neg\phi$. Thus, $\neg AF\neg\phi$ says that there is an infinite path along which ϕ always hold, which is abbreviated by $EG\phi$. For example, if cs_i says that process i is in the critical section, then the formula $EG\neg(\mathsf{cs}_1 \wedge \mathsf{cs}_2)$ says that there is a computation along which we cannot have both process 1 and process 2 in the critical section.

We now consider two algorithmic aspects of CTL. First, there is the model-checking problem, i.e., deciding whether a given CTL-formula holds in a given state in a given finite Kripke structure . Second, there is the validity problem, i.e., deciding if a given CTL-formula holds in all states in all transition systems.

We start with model checking.

Proposition 4.1 [CES86] (cf. [CE81, QS82]) *There is an algorithm that, given a finite Kripke structure M, a state s of M, and a CTL-formula ϕ, determines, in time $O(||M|| \times |\phi|)$, whether $(M, s) \models \phi$.*

Proposition 4.1 is an extension of Proposition 2.1 (since the temporal connective AX correspond to the modality $\Box$). The proof is based on the fact that one can check the satisfaction of AU and EU formulas using graph-based algorithms that run in linear time. As simple as it seems, Proposition 4.1 has an enormous implication for the verification of finite-state programs, having given rise, together with the results in [LP85], to the possibility of *algorithmic verification* of finite-state systems, cf. [CGL93].

We now consider the validity problem. It turns out that Theorem 2.4 applies also to CTL.

Theorem 4.2: [FL79a] *If a CTL-formula ϕ is satisfiable, then ϕ is satisfiable in a Kripke structure with at most $2^{|\phi|}$ states.*

Theorem 4.1 implies the decidability of the validity problem for CTL. Precise lower and upper complexity bounds were established in, respectively, [FL79a] and [EH85].

Theorem 4.3: [FL79a, EH85] *The validity problem for CTL is EXPTIME-complete.*

Is there a "first-order" explanation for the decidability of CTL? As in Section 3, Kripke structures are essentially relational structures over a vocabulary consisting of many unary predicates and one binary predicate. CTL-formulas, however, cannot be viewed as first-order formulas. For example, the CTL-formula EFp holds at a state s if there is a path from s leading to a state in which $p\ holds$. It follows easy from known limitations of the expressive power of first-order logic (cf. [Fag75]) that this is not expressible in first-order logic. CTL, however, can be viewed as a fragment of fixpoint logic, in fact as a fragment of the 2-variable fragment FP^2 of fixpoint logic. (CTL can also be translated into *2-variable transitive-closure logic*, see [IV96].)

We establish this correspondence in two steps. We first define a *modal fixpoint logic*, which is an extension of modal logic with a fixpoint construct. This extension was called in [Koz83] the *propositional μ-calculus*. It is known that CTL can be viewed as a fragment of this logic (see [EC80])). We then observe that modal fixpoint logic can be viewed as a fragment of FP^2.

The syntax of modal fixpoint logic is defined as follows. In addition to the set Φ of propositional constants, we have a set Ψ of *propositional variables*, typically labeled $X, X', Y, Y', \ldots$. The set of modal fixpoint formulas is defined as follows:

- p, for all $p \in \Phi$,
- X, for all $X \in \Psi$,
- $\neg\psi$ or $\psi \wedge \psi'$, where ψ and ψ' are formulas.
- $\Box\psi$, where ψ is a formula.
- $\mu X.\psi$, where ψ is a formula in which X occurs positively (i.e., under an even number of negations).

All occurrence of a propositional variable X within the scope of μX are *bound*; the other occurrences are *free*. A formula in which all propositional variables are bound is called a *closed* formula. We denote by $\phi(X_1, \ldots, X_k)$ a formula all of whose free propositional variables are among $X_1, \ldots, X_k$.

Formulas of the modal fixpoint calculus are interpreted with respect to triples (M, s, V), where $M = (S, \pi, \mathcal{R})$ is a Kripke structure, s is a state of M, and $V : \Psi \rightarrow 2^S$ is a *variable interpretation* that associates with each propositional variable in Ψ a set of states in S (i.e., V is analogous to π). If V is such an interpretation, X is a propositional variable in Ψ, and S' is a subset of S, then $V[X \mapsto S']$ is a variable interpretation that agrees with V on all propositional constants except for X and $V(X) = S'$.

- $(M, s, V) \models p$ (for $p \in \Phi$) if $s \in \pi(p)$.
- $(M, s, V) \models X$ (for $X \in \Psi$) if $s \in V(X)$.
- $(M, s, V) \models \psi \wedge \psi'$ iff $(M, s, V) \models \psi$ and $(M, s, V) \models \psi'$.
- $(M, s, V) \models \neg\psi$ if $(M, s, V) \not\models \psi$.
- $(M, s, V) \models \Box\psi$ if $(M, t, V) \models \psi$ for all t such that $(s, t) \in \mathcal{R}$.
- $(M, s, V) \models \mu X.\psi$ if $s \in \bigcap\{T \subseteq S \mid T = \{t \mid (M, t, V[X \mapsto T]) \models \psi\}\}$

Intuitively, $(M, s, V) \models \mu X.\psi$ if s is in the least fixpoint of ψ, where the latter is viewed as an operator. For a detailed explanation of this perspective, see [Eme96].

It is easy to see that when considering the satisfaction of a formula $\phi(X_1, \ldots, X_k)$, it suffices to consider variable interpretations that are defined on the free propositional variables $X_1, \ldots, X_k$. Thus, if ϕ is a closed formula, then we can talk about ϕ holding in a state s of a Kripke structure s, denoted as usual by $(M, s) \models \phi$.

As mentioned earlier, CTL is subsumed by the modal fixpoint logic. Clearly, AXp and EXp are simply notational variants for $\Box p$ and $\Diamond p$, respectively. More significantly, $A(pUq)$ can be written as $\mu X.q \vee (p \wedge \Box X)$, and $E(pUq)$ can be written as $\mu X.q \vee (p \wedge \Diamond X)$. For a detailed exposition see [Eme96].

It remains to show that modal fixpoint logic can be viewed as a fragment of FP^2. Recall that FP is obtained by augmenting first-order logic with the least-fixpoint operator [CH82]. Let $\phi(\mathbf{x}, S)$ be a formula with free individual variables among $\mathbf{x} = (x_1, \ldots, x_m)$ and in which an m-ary relation symbol S occurs positively (i.e., in the scope of an even number of negations). Over a fixed relational structure, the formula ϕ can be viewed as an operator from m-ary relations to m-ary relations:

$$\phi(P) = \{\mathbf{t} \mid \phi(\mathbf{t}, P) \text{ holds}\}.$$

Because S occurs positively in ϕ, the operator ϕ is monotone, i.e, if $P \subseteq Q$, then $\phi(P) \subseteq \phi(Q)$. Since the (possibly transfinite) sequence

$$\emptyset \subseteq \phi(\emptyset) \subseteq \phi(\phi(\emptyset)) \cdots$$

is increasing, it has a limit, denoted ϕ^∞, which is the least fixpoint of ϕ The formula $\mu S(\mathbf{x}).\phi(\mathbf{x}, S)(\mathbf{z})$ (whose free variables are those in $\mathbf{z}$) refers to this least fixpoint. Formally we have that $\mu S.\phi(\mathbf{x}, S)(\mathbf{t})$ holds in a given relational structure if $\mathbf{t} \in \phi^\infty$.

FP^2 is the fragment of FP obtained by restricting the formulas to refer to only two individual variables, say, x and y. For example, the FP^2-formula $\mu Q.p(x) \vee \exists y(R(x, y) \wedge Q(y))$ describes the set of all nodes in a graph from which one can reach a node where p holds.

We already saw that modal logic can be translated into FO^2. To translate modal fixpoint logic to FP^2, we just have to add one more clause:

- $(\mu P.\phi)^+ = \mu P.\varphi^+$.

Note that we can combine the two translations (from CTL to modal fixpoint logic and from the latter to FP^2) to get a translation from CTL to FP^2. For example, the CTL-formula $E(pUq)$ can be written as $\mu Q.q(x) \vee (p(x) \wedge \exists y(R(x, y) \wedge \exists x(x = y \wedge Q(x)))$.

Theorem 4.4:

(a) $(M, s) \models \phi$ *iff* $(M^*, V) \models \phi^+(x)$, *for each assignment* V *such that* $V(x) = s$.

(b) ϕ *is a valid modal formula iff* ϕ^+ *is a valid* FP^2*-formula.*

Does the fact that CTL can be viewed as a fragment of FP^2 explains its computational properties? Consider first the complexity of evaluating the truth of formulas. By Proposition 4.1, the truth of modal formulas can be evaluated in time that is linear in the size of the structure and in the size of the formula. How hard it is to evaluate the truth of FP^2 formulas?

Proposition 4.5 [Var95] *The problem of evaluating whether an FP^2-formula $\phi(x, y)$ holds in a given relational structure M over a domain D, with respect to an assignment $V : \{x, y\} \to D$ is in NP∩co-NP.*

It is an open problem whether the bound of Proposition 4.5, which is also the best known complexity bound for the model-checking problem for modal fixpoint logic [BVW94, EJS93], can be improved. Even without such an improvement, we can explain the linear bound of Proposition 4.1. A careful analysis (see [EL86]) of the translation of CTL into modal fixpoint logic, shows that we actually need only a small fragment of modal fixpoint logic in which the alternation between fixpoints and negations is limited, which is why this fragment is called *alternation-free* ($L\mu_1$ in [Eme96]). The model-checking problem for alternation-free modal fixpoint logic can be solved in linear time [BVW94, Cle93]. It can be shown that the quadratic upper bound of Proposition 3.3, can be extended to alternation-free FP^2. Thus, viewing CTL as a fragment of FP^2 does explain the tractability of model checking. (For a detailed discussion of the model-checking problem for modal fixpoint logic, see [Eme96].)

It is not clear, however, that viewing CTL as a fragment of FP^2 explains the decidability of the validity problem. The validity problem for modal fixpoint logic is decidable; in fact, like for CTL, it is EXPTIME-complete [EJ88]. One might have expected the validity problem for FP^2 also to be decidable. Unfortunately, it was shown recently in [GOR96] that the validity problem for FP^2 is highly undecidable (Π^1_1-complete), even under various syntactical restrictions. It turns out that logics such as CTL and modal fixpoint logic enjoy an important property, called the *tree-model property*, which does not extend to FP^2.

5 The Tree-Model Property

Consider two Kripke structures $M_1(S_1, \pi_1, \mathcal{R}_1)$ and $M_2(S_2, \pi_2, \mathcal{R}_2)$. Assume $u \in S_1$ and $v \in S_2$. A binary relation $\sim \subseteq S_1 \times S_2$ is a *bisimilarity* relation ([Mil89, Par81]) if the following hold for each pair u, v of nodes such that $(u, v) \in \sim$:

- $\{p \in \Phi \mid u \in \pi_1(p)\} = \{p \in \Phi \mid v \in \pi_2(p)\}$
- If $(u, u') \in R_1$, then there is some $v' \in S_2$ such that $(v, v') \in R_2$ and $u' \sim v'$.
- If $(v, v') \in R_2$, then there is some $u' \in S_1$ such that $(u, u') \in R_2$ and $u' \sim v'$.

Two nodes u, v are *bisimilar*, denoted $u \sim v$, if they are related by some bisimilarity relation. Intuitively, two nodes are bisimilar if they "look alike" locally. It turns out that modal fixpoint logic (and consequently also CTL) cannot distinguish between bisimilar states.

Proposition 5.1 [HM85] *Let $M_1(S_1, \pi_1, \mathcal{R}_1)$ and $M_2(S_2, \pi_2, \mathcal{R}_2)$ be Kripke structures, and assume $u \in S_1$ and $v \in S_2$. If $u \sim v$, then $(M_1, u) \models \phi$ iff $(M_2, v) \models \phi$, for each modal fixpoint formula ϕ.*

An immediate conclusion from Proposition 5.1 is that satisfaction of modal fixpoint formulas is invariant under "unwinding" of Kripke structures. For example, the Kripke structure M' of Figure 2 was obtained from the Kripke structure M of Figure 1 by "unwinding" the node v. It follows that the nodes u in M and the node u in M' are bisimilar. Consequently, they both satisfy the formula $\Box\Box\neg q$.

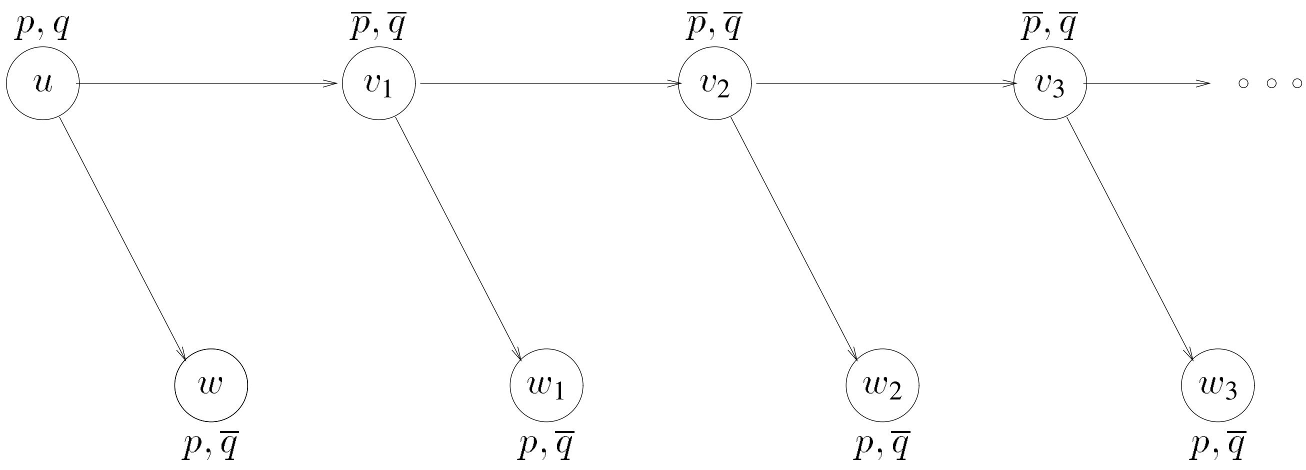

Figure 2: The Kripke structure M'

In particular, Kripke structures can be unwound into *trees*. A tree is a set $T \subseteq \mathcal{N}^*$ (here $\mathcal{N}$ is the set of natural numbers) such that (a) if $x \in T$, then $x \cdot c \in T$ for some $c \in \mathcal{N}$, and (b) if $x \cdot c \in T$, where $x \in \mathcal{N}^*$ and $c \in \mathcal{N}$, then also $x \in T$, and, for all $0 \leq c' < c$, also $x \cdot c' \in T$. The elements of T are called *nodes*, and the empty word ε is the *root* of T. For every $x \in T$, the nodes $x \cdot c \in T$ are the *successors* of x. The number of successors of x is called the *degree* of x and is denoted by $d(x)$. A *branch* of the tree is a sequence $x_0, x_1, \ldots$ such that x_i is a successor of x_{i-1} for all $i > 0$. If all nodes x of T have the same degree $n > 0$, then the tree is called an n-ary tree. Note that in this case $T = \{0, \ldots, n-1\}^*$, denoted T_n.

A Kripke structure $M = (T, \pi, \mathcal{R})$, where T is a tree and $\mathcal{R}$ is the successor relation on T, is called a *tree structure*. It is easy to see that every Kripke structure can be "unwound" to a tree structure (our assumption that the possibility relation is total is important here). Note that the tree structure could be infinite, even when M is finite.

Proposition 5.2 *Let $M = (S, \pi, \mathcal{R})$ be a Kripke structure and assume $u \in S$. Then there is a tree structure $M' = (T, \pi', \mathcal{R}')$ such that $u \sim \varepsilon$.*

It follows from Proposition 5.2 that if a CTL-formula is satisfiable, then it is satisfiable in the root of a tree structure. This is called the *tree-model property*. It should be noted that the tree-model property is incomparable to the finite-model property; there are modal logics whether the former holds, but the latter fails [VW86], while FO^2 has the finite-model property, but not the tree-model property (consider, for example, the FO^2-sentence $\forall x \forall y R(x, y)$).

It turns out that one could prove an even stronger version of the tree-model property for CTL.

Proposition 5.3 [Eme90] *Let ϕ be a CTL-formula. If CTL is satisfiable, then it is satisfiable at the root of a n-ary tree structure, where $n \leq |\phi|$.*

(A similar result can be proven for modal fixpoint logic [SE84], but there is a subtlety regarding the degree that we'd rather not get into here.)

The tree-model property is so important because it provides us with a powerful tool to prove decidability results using Rabin's result about the logic SnS [Rab69]. SnS is a monadic second-order logic about n-ary tree structures. In SnS we view tree structures as relational structures, that is, the nodes of the tree are the elements of the domain and the propositional constants are viewed as unary predicates. We also view the propositional variables as unary predicates. Instead of having, however, one binary predicate, we have n binary predicates $R_0, \ldots, R_{n-1}$, where the interpretation of R_i is the relation $\{(x, x \cdot i) \mid x \in T_n, 0 \leq i < n\}$. Thus, the atomic formulas are either of the form $x = y$, of form $P(x)$, where P is a unary predicate, or of the form $R_i(x, y)$. In addition to Boolean connectives and first-order quantifiers, we allow also monadic second-order quantifiers such as $\exists P$, where P is variable predicate. Intuitively, $\exists P(\phi(P))$ says that there exists a set S' of nodes such that the interpretation $V(P) = S'$ satisfies the formula ϕ. An SnS-formula is *closed* if all its individual variables and all its variable predicates are quantified.

Proposition 5.4 [Rab69] *The validity problem for closed SnS-formulas is decidable.*

We now give a few examples of SnS-formulas. The formula $\bigvee_{i=1}^{n} R_i(x, y)$, abbreviated $succ(x, y)$, says that y is a successor of x. The formula $\forall x \exists y (P(x) \rightarrow succ(x, y) \wedge P(y))$, abbreviated $down(P)$, says that P is in some sense downward closed, every node that satisfies P has a successor that satisfies P. The formula $\forall x \forall y \forall z (P(x) \wedge succ(x, y) \wedge succ(x, z) \wedge P(y) \wedge P(z) \rightarrow y = z)$, abbreviated $unique(P)$, says that a node in P has at most one successor in P. The formula $down(P) \wedge unique(P)$, abbreviated $path(P)$, says that P corresponds to a path in the tree, i.e., the interpretation of P is a set $s_0, s_1, \ldots$, where s_i is the successor of s_{i-1} for all $i > 0$. The formula $\forall x (P(x) \rightarrow q(x))$, abbreviated $sat(P, q)$, says that all elements in P satisfy q. Thus, the formula $\exists P (P(x) \wedge path(P) \wedge sat(P, q))$ says that there is a path where q holds along the path. This corresponds to the CTL-formula EGq.

More generally, every CTL-formula can be expressed as an SnS formula. To express the CTL-formula $A(pUq)$ we proceed as follows. Let $top(Q, x)$ be the formula $Q(x) \wedge \forall y (succ(y, x) \rightarrow \neg Q(y))$. We can now express the CTL-formula $A(pUq)$ by the following SnS formula:

$$
\begin{aligned}
\exists P \exists Q (P(x) \wedge path(P) \wedge path(Q) \wedge sat(Q, P) \\
\wedge \exists x Q(x) \wedge \forall x (P(x) \wedge \neg Q(x) \rightarrow p(x)) \\
\wedge \forall x (top(Q, x) \rightarrow q(x))).
\end{aligned}
$$

Since CTL can be expressed in SnS and the validity problem for SnS is decidable, it follows, by Proposition 5.3, that the validity problem for CTL is decidable. It can be similarly shown that modal fixpoint formulas can be expressed in SnS, yielding the decidability of the validity problem for modal fixpoint logic [KP84]. Thus, SnS provides us with a general framework for proving decidability results for modal logics [Gab73, Gab75].

Unfortunately, the reduction to SnS is not too useful, since the validity problem for SnS is *nonelementary* (that is, there is a lower bound on the time complexity of the form

$$2^{\cdot^{\cdot^{2^n}}},$$

where the height of the stack is n) [Mey75].

A strategy to get around this difficulty was conceived by Streett [Str82]. Streett observed that the crux of Rabin's decidability proof for SnS is its reduction to an automata-theoretic problem. Rabin observed that a tree structure can be viewed as a *labeled tree*. A Σ-labeled tree, for an alphabet Σ is a pair (T, σ), where T is a tree and $\sigma : T \rightarrow \Sigma$ associates with every node of T a label in Σ. A tree structure $(T, \pi, \mathcal{R})$ can be viewed as a 2^{Φ}-labeled tree (T, σ_π), where $\sigma_\pi(x) = \{p \mid x \in \pi(p)\}$, i.e., the label of a node is the set of propositional constants that hold in that node. Rabin showed that with each closed SnS-formula ϕ one can effectively associate a tree automaton A_ϕ such that ϕ holds in an n-ary tree structure $(T_n, \pi, \mathcal{R})$ precisely when A_ϕ accepts the 2^{Φ}-labeled n-ary tree (T_n, σ_π).

An n-ary *tree automaton* A is a tuple $(\Sigma, Q, Q^0, \rho, F)$, where Σ is a finite alphabet, Q is a finite set of states, $Q^0 \subseteq Q$ is a set of initial states, F is an acceptance condition (which will be discussed shortly), and $\rho : Q \times \Sigma \rightarrow 2^{Q^n}$ is a transition function. The automaton A takes as input a Σ-labeled n-ary tree (T_n, σ). Note that $\rho(u, a)$ is a set of n-tuples for each state u and symbol a. Intuitively, when the automaton is in state u and it is reading a node x in T_n, it nondeterministically chooses an n-tuple $\langle u_1, \ldots, u_n \rangle$ in $\rho(u, \sigma(x))$ and then makes n copies of itself and moves to the node $x \cdot i$ in the state u_i for $i = 1, \ldots, n$. A *run* $r : T_n \rightarrow Q$ of A on (T_n, σ) is an Q-labeled n-ary tree such that the root is labeled by an initial state and the transitions obey the transition function ρ; that is, $r(\varepsilon) \in Q^0$, and for each node x we have $\langle r(x \cdot 1), \ldots, r(x \cdot n) \rangle \in \rho(r(x), \sigma(x))$. The run is *accepting* if r satisfies the acceptance condition F. The *language* of A, denoted $L(A)$, is the set of trees accepted by A.

Since, as mentioned above, ϕ holds in an n-ary tree structure $(T_n, \pi, \mathcal{R})$ precisely when A_ϕ accepts the 2^{Φ}-labeled n-ary tree (T_n, σ_π), it follows that ϕ is valid iff $L(A_{\neg\phi}) = \emptyset$. Thus, the validity problem is reduced to the *emptiness problem* for tree automata, i.e., the problem of determining whether the language accepted by a given tree automaton is empty. Streett's insight was to propose applying this strategy to a logic without going through the intermediate step of reducing it to SnS first.

This strategy was carried out for CTL in [VW86] (actually, it was carried out for a closely related logic called PDL [FL79b]), who used *Büchi* tree automata. A Büchi tree automaton is a tree automaton $A = (\Sigma, Q, Q^0, \rho, F)$, where the acceptance condition $F \subseteq Q$ is a set of *accepting* states. A *run* $r : T_n \rightarrow Q$ of A on a Σ-labeled n-ary tree (T_n, σ) is *accepting* if every branch of r visits F infinitely often, that is, if for every branch $x_0, x_1, \ldots$ of T_n there are infinitely many i's such that $r(x_i) \in F$.

Theorem 5.5: *With each CTL-formula ϕ and $n > 0$ one can effectively associate a Büchi tree automaton A_ϕ such that ϕ holds in an n-ary tree structure $(T_n, \pi, \mathcal{R})$ precisely when A_ϕ accepts the 2^{Φ}-labeled n-ary tree (T_n, σ_π). Furthermore, the number of states of A_ϕ is at most exponential in $|\phi|$.*

Theorem 5.5 reduces the validity problem for CTL to the emptiness problem for Büchi tree automata. Rabin described a cubic-time algorithm for emptiness of Büchi tree automata [Rab70]. An improved algorithm with a quadratic running time was described in [VW86]. Combined with Theorem 5.5, this provides an alternative proof for the exponential upper bound in Theorem 4.3. A similar approach, though technically quite more involved, can be used to show that the validity problem for modal fixpoint logic is also decidable in exponential time [EJ88]. It is an open question whether there is a fragment of FP^2 (defined, perhaps, in terms of bounded quantification as in [ABN95]) that is strictly more expressive than modal fixpoint logic but whose validity problem is still decidable. If such a fragment is found, it is likely that the decision procedure for it will be automata-based.

6 In Conclusion

We described the robust tractability of two computational problems associated with modal logic: the model-checking problem and the validity problem. We then asked why, in view of the fact that it is essentially a fragment of first-order logic, modal logic is so robustly decidable. In an attempt, to answer this question, we took a closer look at modal logic as a fragment of first-order logic and noted that it is in fact a fragment of FO^2, which is a tractable fragment of first-order logic. Upon a deep examination, however, we noted that this explanation is not satisfactory. The tractability of modal logic is quite robust and holds also for extensions that are not fragments of FO^2 and are not even first-order. We concluded by showing that modal logic and its extensions enjoy the tree-model property, which leads to automata-based decision procedures.

Acknowledgements. I am grateful to Dov Gabbay, Allen Emerson Neil Immerman, Phokion Kolaitis, and Orna Kupferman, for their comments on previous drafts of this paper.

References

[ABN95] H. Andrika, J. F. A. K. van Benthem, and I. Nimeti. Back and forth between modal logic and classical logic. *J. of the IGPL*, 3:685–720, 1995.

[BAN88] M. Burrows, M. Abadi, and R. Needham. Authetication: a practical study in belief and action. In *Proc. 2nd Conf. on Theoretical Aspects of Reasoning about Knowledge*, pages 325–342, 1988.

[Bay58] A. Bayart. La correction de la logique modale du premier et second ordre S5. *Logique et Analyse*, 1:28–44, 1958.

[Ben74] J. F. A. K. van Benthem. Some correspondence results in modal logic. Report 74–05, University of Amsterdam, 1974.

[Ben85] J. F. A. K. van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Naples, 1985.

[Ben91] J. F. A. K. van Benthem. Temporal logic. Report x-91-05, Institute for Logic, Language, and Computation, University of Amsterdam, 1991.

[BLMS94] R. Brafman, J.-C. Latombe, Y. Moses, and Y. Shoham. Knowledge as a tool in motion planning under uncertainty. In R. Fagin, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. Fifth Conference*, pages 208–224. Morgan Kaufmann, San Francisco, Calif., 1994.

[Boc82] G. V. Bochmann. Hardware specification with temporal logic: an example. *IEEE Transactions on Computers*, C-31:223–231, 1982.

[BVW94] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In D. L. Dill, editor, *Computer Aided Verification, Proc. 6th Int. Conference*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155, Stanford, June 1994. Springer-Verlag, Berlin.

[Car46] R. Carnap. Modalities and quantification. *Journal of Symbolic Logic*, 11:33–64, 1946.

[Car47] R. Carnap. *Meaning and Necessity*. University of Chicago Press, Chicago, 1947.

[CCF82] J. M. V. Castilho, M. A. Casanova, and A. L. Furtado. A temporal framework for database specification. In *Proc. 8th Int. Conf. on Very Large Data Bases*, pages 280–291, 1982.

[CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, January 1986.

[CGL93] E.M. Clarke, O. Grumberg, and D. Long. Verification tools for finite-state concurrent systems. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Decade of Concurrency – Reflections and Perspectives (Proceedings of REX School)*, Lecture Notes in Computer Science, pages 124–175. Springer-Verlag, 1993.

[CH82] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.

[Che80] B. F. Chellas. *Modal Logic*. Cambridge University Press, Cambridge, U.K., 1980.

[Cle93] R. Cleaveland. A linear-time model-checking algorithm for the alternation-free modal μ-calculus. *Formal Methods in System Design*, 2:121–147, 1993.

[CM77] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proc. 9th ACM Symp. on Theory of Computing*, pages 77–90, 1977.

[DG79] D. Dreben and W. D. Goldfarb. *The Decision Problem: Solvable Classes of Quantificational Formulas*. Addison-Wesley, 1979.

[EC80] E.A. Emerson and E.M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. 7th Int'l Colloq. on Automata, Languages and Programming*, pages 169–181, 1980.

[EH85] E.A. Emerson and J.Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.

[EJ88] E.A. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, White Plains, October 1988.

[EJS93] E.A. Emerson, C. Jutla, and A.P. Sistla. On model-checking for fragments of μ-calculus. In *Computer Aided Verification, Proc. 5th Int. Workshop*, volume 697, pages 385–396, Elounda, Crete, June 1993. Lecture Notes in Computer Science, Springer-Verlag.

[EL86] E.A. Emerson and C.-L. Lei. Efficient model checking in fragments of the proposoitional mu-calculus. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 267–278, Cambridge, June 1986.

[Eme90] E.A. Emerson. Temporal and modal logic. *Handbook of theoretical computer science*, pages 997–1072, 1990.

[Eme96] E. A. Emerson. Model checking and the μ-calculus. this volume, 1996.

[Fag75] R. Fagin. Monadic generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:89–96, 1975.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 1995.

[FL79a] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[FL79b] M.J. Fischer and R.E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and Systems Sciences*, 18:194–211, 1979.

[Für81] M. Fürer. The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems). In *Lecture Notes in Computer Science 171*, pages 312–319. Springer-Verlag, 1981.

[Gab71] D. Gabbay. Expressive functional completeness in tense logic. In U. Mönnich, editor, *Aspects of Philosophical Logic*, pages 91–117. Reidel, 1971.

[Gab73] D. Gabbay. A survey of decidability results for modal tense and intermediate logics. In P. Suppes et al., editor, *Proc. 4th Int'l Congress on Logic, Methodology and Philosophy of Science*, pages 29–43. North-Holland, 1973.

[Gab75] D. Gabbay. Decidability results in non-classical logics I. *Annals of Mathematical Logic*, 8:237–295, 1975.

[Gar77] J. W. Garson. Quantification in modal logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Vol. II*, pages 249–307. Reidel, Dordrecht, Netherlands, 1977.

[GKV96] E. Grädel, Ph. G. Kolaitis, and M. Y. Vardi. The decision problem for 2-variable first-order logic. Unpublished manuscript, 1996.

[Göd32] K. Gödel. Ein spezialfall des entscheidungsproblem der theoretischen logik. *Ergebn. math Kolloq.*, 2:27–28, 1932.

[Gol84] W. D. Goldfarb. The Gödel class with equality is unsolvable. *Bull. Amer. Math. Soc. (New Series)*, 10:113–115, 1984.

[GOR96] E. Grädel, M. Otto, and E. Rosen. Undecidability results for two-variable logics. Unpublished manuscript, 1996.

[HC68] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen, London, 1968.

[Hin57] J. Hintikka. Quantifiers in deontic logic. *Societas Scientiarumi Fennica, Commentationes Humanarum Literarum*, 23(4):1–23, 1957.

[Hin61] J. Hintikka. Modalities and quantification. *Theoria*, 27(61):119–128, 1961.

[Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, N.Y., 1962.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of ACM*, 32:137–161, 1985.

[HM90] 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.

[HM92] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[HV89] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time, I: lower bounds. *Journal of Computer and System Sciences*, 38(1):195–237, 1989.

[Imm82] N. Immerman. Upper and lower bounds for first-order expressibility. *Journal of Computer and System Sciences*, 25:76–98, 1982.

[IV96] N. Immerman and M.Y. Vardi. Model checking and transitive closure logic. forthcoming, 1996.

[Kal33] L. Kalmár. Über die erfüllbarkeit derjenigen zälhausdrücke, welche in der normalform zwei benachbarte allzeichen enthalten. *Math. Annal.*, 108:466–484, 1933.

[Kan57] S. Kanger. *Provability in Logic*. Stockholm Studies in Philosophy I, 1957.

[Kel76] R.M. Keller. Formal verification of parallel programs. *Comm ACM*, 19:371–384, 1976.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KP84] D. Kozen and R. Parikh. A decision procedure for the propositional mu-calculus. In *Logics of Programs*, volume 164 of *Lecture Notes in Computer Science*, pages 313–325. Springer-Verlag, 1984.

[Kri59] S. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24:1–14, 1959.

[Kri63] S. Kripke. A semantical analysis of modal logic I: normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963. Announced in (1959) *Journal of Symbolic Logic 24*, 323.

[Lad77] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

[Lam80] L. Lamport. "Sometimes" is sometimes "not never": on the temporal logic of programs. In *Proc. 7th ACM Symp. on Principles of Programming Languages*, pages 164–185, 1980.

[Lew79] H. R. Lewis. *Unsolvable Classes of Quantificational Formulas*. Addison-Wesley, 1979.

[Lip77] W. Lipski. On the logic of incomplete information. In *Proc. 6th International Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, Vol. 53, pages 374–381. Springer-Verlag, Berlin/New York, 1977.

[Liu89] M.T. Liu. Protocol engineering. *Advances in Computing*, 29:79–195, 1989.

[LL59] C. I. Lewis and C. H. Langford. *Symbolic Logic*. Dover, New York, 2nd edition, 1959.

[LP85] 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.

[LR86] R. E. Ladner and J. H. Reif. The logic of distributed protocols (preliminary report). In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pages 207–222. Morgan Kaufmann, San Francisco, Calif., 1986.

[Mer56] C. A. Meredith. Interpretations of different modal logics in the "property calculus", mimeographed manuscript, Philosophy Department, Canterbury University College (recorded and expanded by A. N. Prior). 1956.

[Mey75] A. R. Meyer. Weak monadic second order theory of successor is not elementary recursive. In *Proc. Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 132–154. Springer-Verlag, 1975.

[MH69] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. Michie, editor, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[Mil89] R. Milner. *Communication and Concurrecny*. Prentice-Hall, Englewood Clifs, 1989.

[Mon60] R. Montague. Logical necessity, physical necessity, ethics, and quantifiers. *Inquiry*, 4:259–269, 1960.

[Mor75] M. Mortimer. On language with two variables. *Zeit. für Math. Logik und Grund. der Math.*, 21:135–140, 1975.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, Berlin, January 1992.

[Par81] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proc. 5th GI Conf. on Theoretical Computer Science*, Lecture Notes in Computer Science, Vol. 104. Springer-Verlag, Berlin/New York, 1981.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on Foundation of Computer Science*, pages 46–57, 1977.

[Pra76] V. R. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proc. 17th IEEE Symp. on Foundations of Computer Science*, pages 109–121, 1976.

[Pri62] A. N. Prior. Possible worlds (idea attributed to P. T. Geach). *Philosophical Quarterly*, 12:36–43, 1962.

[QS82] 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.

[Rab69] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Rab70] M.O. Rabin. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, pages 1–23. North Holland, 1970.

[RS83] J. H. Reif and A. P. Sistla. A multiprocessor network logic with temporal and spatial modalities. In *Proc. 12th International Colloq. on Automata, Languages, and Programming*, Lecture Notes in Computer Science, Vol. 104. Springer-Verlag, Berlin/New York, 1983.

[Rud87] H. Rudin. Network protocols and tools to help produce them. *Annual Review of Computer Science*, 2:291–316, 1987.

[Sch34] K. Schütte. Untersuchungen zum entscheidungsproblem der mathematischen logik. *Math. Annal.*, 109:572–603, 1934.

[Sco62] D. Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:377, 1962.

[SE84] R. S. Street and E. A. Emerson. An elementary decision procedure for the mu-calculus. In *Proc. 11th Int. Colloquium on Automata, Languages and Programming*, volume 172. Lecture Notes in Computer Science, Springer-Verlag, July 1984.

[Str82] R.S. Streett. Propositional dynamic logic of looping and converse. *Information and Control*, 54:121–141, 1982.

[Var89] M. Y. Vardi. On the complexity of epistemic reasoning. In *Proc. 4th IEEE Symp. on Logic in Computer Science*, pages 243–252, 1989.

[Var95] M.Y. Vardi. On the complexity of bounded-variable queries. In *Proceedings of the ACM 14th Symposium on Principles of Database Systems*, June 1995.

[VW86] 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.