Why are modal logics so robustly decidable?

Erich Grädel*

Abstract

Modal logics are widely used in a number of areas in computer science, in particular for the specification and verification of hardware and software systems, for knowledge representation, in databases, and in artificial intelligence. The most important reason for the successful applications of these logics is that they provide a good balance between expressive power and computational complexity. In [31] Vardi adressed the question to identify the main reasons for the robust decidability properties of modal logics. In particular he stressed the importance of the tree model property, which permits the use of powerful automata-theoretic techniques. Here we discuss this question in the light of recent research on *guarded fragments* of first-order logic and fixed point logic.

1 Introduction

Quisani: Yuri[1] told me you wanted to see me.

Author: Yes. I am writing a contribution to Yuri's column in the EATCS-Bulletin. Would you like to help me?

Q: It depends. What is the topic this time?

A: The robust decidability properties of modal logic.

Q: Modal logics? Aren't that these weird languages about necessicity and possibility, where you put symbols like $\Box$ and $\Diamond$ in front of other formulae, and that have those cryptic names like K, T, S4, S5 and so on? I always had the impression that this is mainly a subject of philosophers and proof theorists.

A: This is no longer the case. Modal logics are now widely used in many areas of computer science. It is true that in older texts on modal logics there is an overemphasis on syntactic, proof-theoretic issues. But nowadays *"modal languages are viewed as a tool to analyse the properties of labelled transition systems, not the other way round"*, and the analysis of the expressive power of modal logics for concrete computer science scenarios and the investigation of *algorithmic issues* play an important role. To get acquainted with modal logic, I recommend the book by Sally Popcorn [28] (although algorithmic issues are left out there) and the forthcoming book by Blackburn, de Rijke and Venema [6]. Both books advocate a modern view of modal logics.

*Mathematische Grundlagen der Informatik, RWTH Aachen, D-52056 Aachen, Email: graedel@informatik.rwth-aachen.de, URL: www-mgi.informatik.rwth-aachen.de/

[1] Yuri is of course Yuri Gurevich, the former advisor and usual interlocutor of Quisani.

Q: There are so many logical systems around. Why should I use modal logics? Can I say or do more with them than, for instance, with good old first-order logic?

A: In many cases you want to say less! In other cases you just want to say different things than in first-order logic. To be useful for a particular application area (e.g. hardware specification) a logic should provide a good balance between expressive power and algorithmic manageability. This means that on one side the relevant statements for that application area must be expressible in the logic, on the other side, the usual reasoning tasks should be decidable and admit reasonably efficient algorithms. Modal logics (broadly conceived) often provide a good balance, whereas for instance first-order logic in many cases does not.

Q: What are these reasoning tasks?

A: Mainly the *satisfiability* problem (given a formula, decide whether it is true in some model) and the *model checking* or *evaluation* problem (given a formula and a structure, evaluate the formula in that structure). There are other important problems like finite satisfiability, subsumption etc., but these are two most important ones.

Q: And what do you mean by "reasonably efficient algorithms"?

A: This is a delicate point and it is difficult to give a general answer. For model checking problems we would like to have algorithms that are very efficient (say, linear time) with respect to the size of the structure and at least polynomial-time with respect to the size of the formula. In many typical applications, for instance in verification or in databases, the formulae (specifications, queries) tend to be rather small wheras the structures (transition systems, databases) can be enormous.

Q: And for satisfiability problems? Aren't these even undecidable in most cases?

A: For first-order logic, and even for many rather restricted fragments of first-order logic, yes. (You already discussed this with Yuri [22], see also [7]). Fortunately the situation is different for modal logics, and this is one of the reasons why we discuss them here. Still, the complexity of satisfiability problems for interesting logics is rather high. As you know, satisfiability is NP-complete even for plain propositional logic, and for more powerful logic it quickly becomes at least PSPACE- or EXPTIME-hard. For instance, the satisfiability problem for propositional modal logic ML is PSPACE-complete [26]. I once heard Moshe Vardi say in a talk more or less the following: "For many combinatorial problems, quadratic time complexity is too high in practice. But if you have a satisfiability problem in logic and you can solve it with a PSPACE-algorithm, it is a reason for a party!" Surprisingly people have built systems that solve PSPACE- or EXPTIME-complete satisfiability problems of moderate size quite reasonably. We do not really understand, why. It seems hard to find the hard instances, even though we know that there must be many of them. At least they do not seem to occur naturally in many applications.

Q: What do you mean by 'robust decidability properties'?

A: The basic propositional (poly-)modal logic ML — or K_n, as it is sometimes called by modal logicians — is decidable, and remains so even if one adds features like number restrictions, path quantification, least and greatest fixed points and so on, which are of crucial importance for applications ...

Q: You are going too fast! Could you please remind me what basic modal logic precisely means.

2 Modal logic

A: The semantic objects that modal logics talk about are *transition systems*, which really are just relational structures that have only unary and binary predicates. For historical reasons, they are also called *Kripke structures.* A transition system K consists of a (finite or infinite) universe whose elements we call *states*, a (finite) collection of unary predicates P_i which are called *atomic propositions* P_i and a (finite) collection $(E_a)_{a \in A}$ of binary *transition relations.*

ML-formulae are built from the atomic propositions P_i, using the boolean connectives, and the modal operators $\langle a \rangle$ and $[a]$ which are used as follows. For each formula ψ and each 'action' $a \in A$ we can build new formulae $\langle a \rangle \psi$ and $[a]\psi$. A simple example of an ML-formula is $P_1 \wedge \langle a \rangle (P_2 \wedge [b]P_1)$.

Q: Can you tell me how ML-formulae are evaluated on a transition system?

A: A modal formula is always evaluated at a particular state of the transition system. We write $K, v \models \psi$ to denote that ψ is true at state v of the transition system K. For atomic propositions, we have $K, v \models P_i$ iff $v \in P_i$ (atomic propostions are identified with the set of states at which they are true), the meaning of the boolean connectives is the usual one, and the modal operators are interpreted as follows: $K, v \models \langle a \rangle \psi$ iff ψ is true at some a-successor of v, i.e., iff there exists a state w such that $vE_a w$ and $K, w \models \psi$. Dually, $K, v \models [a]\psi$ iff ψ is true at *all* a-successors w of v in K.

Q: Please give me an example.

A: The sentence $P_1 \wedge \langle a \rangle (P_2 \wedge [b]P_1)$ is true at a state v if P_1 is true at v, and there is an a-transition from v to a state w where P_2 holds and from which every b-transition leads to a state where again P_1 holds.

Q: But frankly, this seems to be a very weak logic. With an ML-formula I can only look from the current state along a bounded number of transition steps. I don't see how this can be sufficient to reason about computations.

A: You are right, ML is very weak. Some of its limitations are:

- It does not have equality, so you cannot say that an a-transition from the current state brings you to the same state as a b-transition.
- It does not have unbounded quantification, so you cannot say that something holds at all states.
- In ML you cannot define new transition relations, not even on the quantifier-free level. For instance, you cannot say that the current state is reachable from some other state w by both an a-transition and a b-transition (because there is no way to define the new transition $Exy \equiv E_a yx \wedge E_b yx$).
- ML has no counting mechanism whatsoever. For instance you cannot say that there is at most one a-transition from the current state, so you cannot distinguish nondeterministic from deterministic systems.
- Perhaps the most important limitation is the one you just alluded to, namely that there is no recursion mechanism in ML. But note that contrary to the other limitations, this is also a limitation of first-order logic.

Q: So ML is really too weak for interesting applications. Then why did you say that modal logics are useful?

A: There are powerful extensions of modal logics that are expressive enough for computationally interesting properties and still algorithmically manageable. Actually many of these logics are of considerable practical importance ...

Q: Such as?

A: ... such as CTL (computation tree logic) [10] and the μ-calculus [25]. Another interesting class of such languages are the so-called *description logics* [9] used in knowledge representation (to mention a different application area). These are also basically modal logic, although this is not immediately apparent from their 'official' definition.

Q: Can you explain one or two of these logics, please?

A: Let us just look at some examples for CTL and the μ-calculus. CTL extends ML by path quantifiers, and by temporal operators like **next**, **eventually** and **until** on paths. So you can express statements about infinite computation paths in a transition system, such as "on all paths, eventually φ" or "never, a deadlock state can be reached" and so on. The μ-calculus is an even more powerful logic than CTL, that encompasses also many other logics such as PDL, LTL and CTL*. It extends ML by least and greatest fixed points. For instance, the formula $\mu X.\varphi \vee [a]X$ says that the current state is in the least fixed point of the operator $X \mapsto \{v : K, v \models \varphi \vee [a]X\}$. Do you see what this means?

Q: The operator maps X to the set of states where either φ is true or all a-transitions lead to a state that belongs to X. So the least fixed point is the set of states where all a-paths eventually hit a state where φ holds.

A: Good. Another examples is $\nu X. \bigvee_{a\in A}\langle a\rangle \mathit{true} \wedge \bigwedge_{a\in A}[a]X$. Here ν stands for the greatest fixed point. This formula says that the current state is in the greatest fixed point of the operator that assigns to X the set of states that are not deadlock states (i.e. have at least one outgoing transition) and where all outgoing transitions lead to a state in X. Hence the formula says that from the current state no deadlock will ever be reached.

Q: I see that one can say relevant things with these logics. But are they really still algorithmically manageable.

A: Model checking is very fast (i.e. linear time both in the size of the transition system and the length of the formula) for CTL and, more generally, for such formulae of the μ-calculus that do not contain alternations of least and greatest fixed points [8]. For formulae with an unbounded number of alternations we do not know whether polynomial-time model checking is possible (we know that it is in NP $\cap$ Co-NP). This is a very interesting open problem, at least theoretically. In practice we almost never need formulae with more than two alternations, and for these we have good model checking algorithms (see [11] for a survey).

Q: And satisfiability?

A: The satisfiability problems for CTL and the μ-calculus are EXPTIME-complete [12, 13] (compared to PSPACE for ML). By the way, both logics also have the *finite model property*: Every satisfiable sentence has a finite model [24]. There is by now a huge literature on the μ-calculus and CTL.

3 Modal logic and first-order logic

Q: Let me get back to the basic logic ML. What puzzles me is that while it syntactically looks like a propositional logic, the semantics is not provided by assignments of truth-values to propositional variables but by structures in the sense of Tarski.

A: You can have a 'local' view of a modal formula ψ or a 'global' one. In the local view, you imagine yourself sitting inside the structure at a specific state (or world) and you ask whether ψ is true at the place where you sit. Hence you consider modal formulae as propositions that are either true or false, and the 'modalities' are actions that take you from one place in your structures to another one. From this local point of view, you can think of ML as a propositional logic. There are people who claim that users think propositionally and do not like variables, and that therefore modal logics and description logics are more 'user friendly' than other systems.

Q: You said there is also a global view.

A: In the global view you look from outside at the entire Kripke structure and you want to know the set of states at which ψ holds. Hence you consider atomic propositions as unary relations (i.e. subsets) of your structure, and actions as binary relations that describe sets of possible transitions. And each modal formula defines a unary relation, namely the set of states at which it is true.

Q: This looks to me more like first-order logic.

A: Indeed ML should really be seen as a fragment of first-order logic (FO). There is a standard translation, taking every modal formula $\psi \in$ ML to a first-order formula $\psi^*(x)$ with one free variable such that $K, v \models \psi$ iff $K \models \psi^*(v)$.

Q: Oh yes, I see. An atomic proposition P_i is translated into $P_i x$ and the modal operators correspond to quantifiers. The meaning of $\langle a \rangle \psi$ is $\exists y(E_a xy \wedge \psi^*(y))$ and the formula $[a]\psi$ says the same as $\forall y(E_a xy \rightarrow \psi^*(y))$.

A: Very good. The first-order formulae that correspond to ML-formulae under this translation form what van Benthem has called the *modal fragment* of first-order logic. He has shown that it has some very interesting properties. In particular it has a semantic characterization via bisimulation.

Q: What is bisimulation?

A: Bisimulations relate pairs (K, v) and (K', v') of Kripke structures with a distinguished current state. Intuitively (K, v) and (K', v') are bisimilar, if processes in K from v and processes in K' from v' can simulate each other in a strong sense. More formally it means that there exists a bisimulation relation Z between the states of K and the states of K' such that vZv' and whenever uZu' holds then

(1) u and u' are not distinguished by atomic propositions,

(2) the following back and forth conditions are satisfied:

forth: for every $a \in A$ and any transition $uE_a w$ in K there exists a transition $u'E_a w'$ in K' such that wZw'.

back: Similarly for every transition $u'E_a w'$ in K'.

Q: This looks to me like a version of these Ehrenfeucht-Fraïssé games that Yuri and other finite model theorists are always using.

A: You are right. Bisimulation can be viewed as the adaptation of Ehrenfeucht-Fraïssé games to the modal case.

Q: But what precisely has bisimulation to do with modal logic?

A: If (K, v) and (K', v') are bisimilar, then they cannot be distinguished by any modal formula. In fact modal logic is exactly the bisimulation-invariant fragment of first-order logic [3].

Theorem (van Benthem). *The properties definable in* ML *are precisely the properties that are first-order definable and invariant under bisimulation.*

Recently, a similar characterization has been given for the μ-calculus [23].

Theorem (Janin, Walukiewicz). *The properties definable in the μ-calculus are precisely the properties that are definable in monadic second-order logic and are invariant under bisimulation.*

4 Two-variable logics

Q: Looking more closely at the standard translation from ML into FO, I notice that I only need a very small part of first-order logic. Suppose that I translate $\langle a\rangle\psi$ into $(\langle a\rangle\psi)^*(x) := \exists y(E_a xy \wedge \psi^*(y))$. Then, inside the construction of $\psi^*(y)$, when I come to the translation of the next modal operator, I can re-use x and do not need a new variable. So the entire translation can be done with only two variables.

A: Very good. Indeed it has been noticed already in the 1970s that ML can be embedded into FO^2, the *two-variable fragment* of first-order logic. Traditionally this has been proposed as an explanation for the good properties of modal logic. But unfortunately it has turned out that this explanation is not really convincing.

Q: Why not? Is FO^2 not decidable?

A: Yes, it is. Mortimer has proved this already in 1975 [27]. He has also shown that FO^2 has the finite model property. For a simpler proof of these results (with optimal complexity bounds) see [17].

Q: So what is the problem?

A: Actually there are a number of problems. First, FO^2 is more complex. While the satisfiability of ML-formulae is in PSPACE, the same problem for FO^2 is NEXPTIME-complete. ML has a number of nice model-theoretic properties that fail for FO^2 [1, 20]. Perhaps most importantly, FO^2 is not nearly as robustly decidable as modal logic.

Q: What does this mean?

A: Look at logics that extend FO^2 in a similar way as CTL or the μ-calculus extend basic modal logic. For instance one can add transitive closures, a variant of path quantification, or least and greatest fixed point operators to FO^2. But even if one does this in the most restrictive conceivable way, the resulting logics become undecidable, in fact undecidable of very high degree (Σ^1_1-hard) [18, 19].

5 The tree model property

Q: Can you give me an intuitive reason why modal logic remains decidable when you add things like path quantification or fixed points, and FO^2 does not.

A: As has been pointed out by Moshe Vardi [31], one of the main reasons is that modal logics have the so-called *tree model property*.

Q: What does this mean? Something like the finite model property?

A: It means that every satisfiable formula has a model that is a tree. In fact, there even is a tree model with bounded branching. So yes, it is somewhat similar to the finite model property. Surprisingly, the tree model property often seems to be even more useful than the finite model property for obtaining good complexity results and for the design of practical algorithms (which is not the same thing).

Q: But why do modal logics have the tree model property?

A: If you do not insist on bounded branching of the underlying tree, this follows very easily from the closure under bisimulation. Take a Kripke structure K that satisfies your formula ψ at some state v. Then build a tree K' by 'unraveling' K from v. That is, consider the tree of all finite paths through K that start at v, and consider the end-point of each finite path as a separate node (so when two paths in K from v lead to the same point w, then there will be two copies of w in K' and two different paths never meet.

Q: I see. The unraveled structure at its root is bisimilar to the original structure at state v. Hence ψ is satisfied at the root of the tree structure. And FO^2 does not have the tree model property?

A: No, of course not.

Q: Oh, I see. One can just take the sentence $\forall x \forall y Exy$.

A: Yes. And the extensions of FO^2 that I mentioned are powerful enough to axiomatize grids and therefore to encode undecidable tiling problems or halting problems [18].

A: But why do you say that the tree model property is more important than the finite model property? At least the finite model property always implies decidability ...

A: Well, almost always.

Q: ... but I don't see why this should be the case for the tree model property.

A: The tree model gives us powerful tools for proving decidability results and for constructing efficient decision procedure. For instance we can reformulate μ-calculus formulae as monadic second-order formulae and then use Rabin's famous theorem saying that $S\omega S$, the monadic second-order theory of the ω-branching tree, is decidable (see [29] for the original proof and [7, Chapt. 7] for a modern exposition).

Q: Wait. If my recollection is right then the complexity of these monadic theories is absolutely awful. More than any bounded number of iterations of the exponential function.

A: Indeed. The reduction to Rabin's Theorem just gives a simple decidability proof, but no good complexity results and certainly no practical decision procedures.

Q: So how can you do better?

A: The proof of Rabin's Theorem is based on tree automata. By using automata directly for the μ-calculus, we indeed get good algorithms, both for model checking problems and for satisfiability problems [2, 14].

Q: So does the tree model property answer the question why modal logics are robustly decidable?

A: It is a partial answer, yes [31]. But we would like to generalize it. We would like to have more powerful logics than ML, CTL and even the μ-calculus that retain the nice properties of modal logics.

6 Guarded logics

A: Let us look again at the standard translation from modal logic into first-order logic. Apart from the fact that it never needs more than two variables, what else is special.

Q: The quantifiers always have these atoms $E_a xy$ attached to them.

A: Exactly. The quantifiers are always relativized by atoms. Some years ago, Andréka, van Benthem and Németi [1] suggested that this is really main reason for the good properties of modal logics. So they removed all other restrictions of the modal fragment (the limitation to unary and binary predicates and the restriction to two variables) and considered the fragment of first-order logic where all quantifiers are guarded by atoms. They formulated the 'thesis' that the *guarded fragment* GF of FO generalizes and explains most of the good properties of modal logic.

Q: Can you give me a more precise definition of the guarded fragment.

A: Here it is.

(1) Every relational atomic formula $Rx_{i_1} \cdots x_{i_m}$ or $x_i = x_j$ belongs to GF.

(2) GF is closed under boolean operations.

(3) If $\boldsymbol{x}, \boldsymbol{y}$ are tuples of variables, $\alpha(\boldsymbol{x}, \boldsymbol{y})$ is a positive atomic formula and $\psi(\boldsymbol{x}, \boldsymbol{y})$ is a formula in GF such that all free variables of ψ occur in α, then also the formulae

$$\exists \boldsymbol{y}(\alpha(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y})) \quad \text{and} \quad \forall \boldsymbol{y}(\alpha(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \psi(\boldsymbol{x}, \boldsymbol{y}))$$

belong to GF.

The atom $\alpha(\boldsymbol{x}, \boldsymbol{y})$ is called the *guard* of the quantifier. It is important that the guard contains all the free variables in the scope of the quantifier.

Q: I can see that the standard translation maps basic modal logic into GF. In what sense does GF extend ML.

A: We are not restricted anymore to unary and binary relations. But even on transition systems we now have the possibility to define new transition relations. Further we do not have the strict separation between 'state properties' and transitions that we always have in modal logics.

Q: Can you give me examples that are *not* in GF.

A: Look at the axiom for transitivity. You can write either it in the form $\forall xyz(Exy \land Eyz \rightarrow Exz)$ or $\forall xy(Exy \rightarrow \forall z(Eyz \rightarrow Exz))$. Neither of them is guarded. Another instructive non-example is $\forall y(x \leq y \land y \leq z \rightarrow \varphi(y))$ saying that all points between x and z have a property φ.

Q: But this is really disappointing! Wouldn't you need something like that 'between-formula' if you wanted to translate temporal formulae like (φ **until** ψ) into first-order logic? So, for instance, you cannot embed temporal logic statements over $(\mathbb{N}, <)$ into GF?

A: You are right. Both examples suggest that we would like to have a more general notion of guards that can also be conjunctions of atoms. But one has to be careful, and actually the transitivity axiom and the 'between-formula' behave differently. The quantifier in the 'between-formula' is guarded in a weaker sense, which seems to suffice for most of the good properties of guarded formulae. This motivated van Benthem [4] to define the more general *loosely guarded fragment* LGF, where the quantifier-rule is relaxed as follows:

(3)′ If $\psi(\boldsymbol{x}, \boldsymbol{y})$ is in LGF, and $\alpha(\boldsymbol{x}, \boldsymbol{y}) = \alpha_1 \land \cdots \land \alpha_m$ is a conjunction of atoms, then

$$\exists \boldsymbol{y}((\alpha_1 \land \cdots \land \alpha_m) \land \psi(\boldsymbol{x}, \boldsymbol{y})) \quad \text{and} \quad \forall \boldsymbol{y}((\alpha_1 \land \cdots \land \alpha_m) \rightarrow \psi(\boldsymbol{x}, \boldsymbol{y}))$$

belong to LGF, provided that $\text{free}(\psi) \subseteq \text{free}(\alpha) = \boldsymbol{x} \cup \boldsymbol{y}$ and for any two variables $z \in \boldsymbol{y}$, $z' \in \boldsymbol{x} \cup \boldsymbol{y}$ there is at least one atom α_j that contains both z and z'.

LGF retains almost all of the nice properties of GF. In particular it is also decidable. In fact there are by now a number of further variations of guarded first-order logics such as the clique-guarded fragment [16] and several variants of action-guarded fragments [5]. But let us no go into further details here.

A: What about the transitivity axiom. Neither of the two formulations is loosely guarded. Perhaps one can reformulate it in a different way to make it loosely guarded.

Q: No you can not. In fact if you add transitivity statements to GF you get an undecidable system.

Q: What is the complexity of GF?

A: The satisfiability problem for GF (and also for LGF) is complete for 2EXPTIME, the class of problems solvable in double exponential time [15].

Q: But this is disappointing again! The complexity of GF is even worse than for FO^2. How can Andréka et al. claim that guarded quantification explains the good *algorithmic* properties of modal logic, if ML is PSPACE and GF is 2EXPTIME?

A: This comparison is too superficial and in some sense unfair. The reason for the double exponential complexity of GF is that the arity of the predicate symbols is not bounded, whereas modal logic and FO^2 only speak about graphs. Even a single n-ary predicate on a domain of only two objects gives 2^{2^n} types already on the atomic level. Hence the double exponential lower complexity bound is hardly a surprise. Further, in most of the potential applications of guarded logics the vocabulary is fixed and the arity therefore bounded. But for GF-sentences of bounded arity, the satisfiability problem can be decided in EXPTIME [15].

Q: Still, EXPTIME is higher than PSPACE.

A: This is probably true. But most of the really interesting modal logics are also EXPTIME-complete, for instance CTL or the μ-calculus. In fact you have to add very little to ML to get a logic that is EXPTIME-hard. Already a universal modality suffices (i.e. the possibility to express that some formula holds at *all* states of the given structure). So I think the complexity argument does not settle the question whether the explanation of Andréka, van Benthem and Németi is adequate.

7 Guarded fixed point logic

Q: I accept that for the moment. But there is another challenge. If you add the features of CTL or the μ-calculus to GF, do you get a decidable logic? If yes, what is its complexity?

A: Indeed, I think this is the real test for the 'thesis' of Andréka, van Benthem and Németi. So let us look at guarded fixed point logic, denoted μGF. It is defined by adding to GF the following rule for defining fixed point formulae:

Let W be a k-ary relation variable and let $\boldsymbol{x} = x_1, \dots, x_k$ be a k-tuple of distinct variables. Further, let $\psi(W, \boldsymbol{x})$ be a guarded formula where W appears only positively and not in guards and that contains no free variables other than $\boldsymbol{x}$. Then we can build the formulae

$$[\text{LFP } W\boldsymbol{x} \,.\, \psi](\boldsymbol{x}) \quad \text{and} \quad [\text{GFP } W\boldsymbol{x} \,.\, \psi](\boldsymbol{x}).$$

The semantics is the usual one: The formula $[\text{LFP } W\boldsymbol{x} \,.\, \psi(W, \boldsymbol{x})](\boldsymbol{a})$ is true in a structure $\mathfrak{A}$ (with universe A) if and only if the tuple $\boldsymbol{a}$ is in the least fixed point of the operator

$$\psi^{\mathfrak{A}} : W \mapsto \psi^{\mathfrak{A}}(W) := \{\boldsymbol{a} \in A^k : \mathfrak{A} \models \psi(W, \boldsymbol{a})\}.$$

Similarly for the greatest fixed points.

Q: I find such fixed point sentences hard to read. Can you give me an example.

A: Here is an instructive example of a guarded fixed point sentence.

$$\exists xyFxy \wedge \forall xy(Fxy \rightarrow \exists xFyx) \wedge \forall xy(Fxy \rightarrow [\text{LFP } Wx \,.\, \forall y(Fyx \rightarrow Wy)](x)).$$

Q: Let me try to understand this. The first two conjuncts say that there exists an F-edge and that every F-edge can be extended to an infinite path. The third conjunct says that each point x on an F-edge is in the least fixed point of the operator $W \mapsto \{w :$ all F-predecessors of w are in $W\}$. The least fixed point is the set of points that have only finitely many F-predecessors.

A: Yes. The sentence says that there is an infinite forward F-chain, but no infinite backwards F-chain. This means in particular that F does not cycle.

Q: So this sentence has only infinite models! But this means that guarded fixed point logic does not have the finite model property.

A: Right.

Q: But then I would conjecture that the satisfiability problem for μGF is undecidable.

A: Wrong.

Theorem (Grädel, Walukiewicz). [21] *The satisfiability problem for guarded fixed point logic is decidable and complete for* 2EXPTIME. *For* μGF*-sentences whose relation symbols have bounded arity, the satisfiability problem is* EXPTIME*-complete.*

The proof is based on alternating tree automata. It is shown that for every sentence $\psi \in \mu$GF one can build an automaton that accepts a tableau-representation of a structure if and only if that structure is a model of ψ. In this way the satisfiability problem for μGF is reduced to the emptiness problem for a particular variant of tree automata, a problem that can be shown to be in EXPTIME.

But the intuitive reason for the nice behaviour of μGF is that it has the *tree model property.* (I told you that the tree model property is more relevant than the finite model property).

Q: I don' t understand your claim on the tree model property. Guarded sentences can have predicates of any arity, so we are not talking about graphs any more. How can a model then be a tree?

A: I should have been more precise. I don't mean trees in the strict sense, but structures of *small tree width.* A structure has tree width k if it can be covered by a tree-shaped arrangement of (in general overlapping) substructures of size at most $k+1$. More formally, if there exists a tree T and for every node v of T a substructure $\mathfrak{F}(v) \subseteq \mathfrak{A}$ with no more than $k+1$ elements such that (1) $\mathfrak{A} = \bigcup_{v \in T} \mathfrak{F}(v)$ and (2) for each element a of $\mathfrak{A}$, the set of nodes v such that $a \in \mathfrak{F}(v)$ is connected (i.e. induces a subtree of T).

Q: Why $k+1$ elements and not k?

A: Because we want trees and forests to have tree width 1. Cycles have tree width 2. A simple class of structures with unbounded tree width are finite rectangular grids. Intuitively, the tree width of structure measures how closely the structure ressembles a tree.

Q: I see. To prove that a problem is undecidable you usually encode in it some undecidable tiling problem on grids. If you have a logic with this generalized tree model property then you cannot do this. Your formulae cannot axiomatize grids since these do not have bounded tree width.

A: Exactly. Further, structures of bounded tree width can be interpreted in trees with a finite set of labels. Then again one can use Rabin's Theorem to prove decidability, and tree automata to get good complexity bounds.

Q: How can one see that guarded fixed point logic has the tree model property?

A: Again by generalizing the methodology from modal logic. Recall that modal logics have the tree model property because they are invariant under bisimulations and because every transition system can be unravelled to a bisimilar tree. The notion of bisimulations can be generalized to guarded bisimulations. In [1] it is shown that the properties expressible in GF are precisely the first-order definable properties that invariant under guarded bisimulations. Also guarded fixed point logic is invariant under guarded bisimulations. Then one can proceed as in the modal case and 'unravel' a structure in this case not to a tree but to a structure of small tree width. There is a guarded bisimulation between a structure and its unraveling. Hence if there is any model of a guarded fixed point sentence then there also is one of small tree width.

Q: But don't you need trees of bounded branching to apply tree automata.

A: For most of the common variants in the literature, yes. That is why also most of the automata-based proofs of decidability and complexity results for the μ-calculus first need to establish the stronger variant of the tree model property, with bounded branching of the underlying tree. This is not easy, it requires a careful analysis of the regeneration sequences of fixed points (see e.g. [30]). Fortunately this is no longer necessary, neither for the μ-calculus, nor for guarded fixed point logic.

Q: Why not?

A: One can actually use alternating automata on arbitrary branching trees. The automata treat all the edges to children of the current node (and in fact also the edge backwards to the parent node) in the same way. A general forgetful determinacy theorem for graph games can then be used to show that if such an automaton accepts a tree then it also accepts one of bounded branching. So the branching property of the trees falls out of general theorems on automata and games, and it is not necessary to prove a bounded branching property *before* you can apply automata. For details, see [21] and a forthcoming paper by Thomas Wilke.

Q: So do you agree with the explanation proposed by Andréka, van Benthem and Németi.

A: Yes. I think that the positive decidability results on guarded logics confirm that the guarded nature of quantification in modal logics is the main responsable for their good algorithmic properties.

But our time is running out. The deadline that Yuri has set me for this article is NOW.

Q: Then you better send it to him!

A: Thank you for your help. Bye.

References

[1] H. Andréka, J. van Benthem, and I. Németi, *Modal languages and bounded fragments of predicate logic*, Journal of Philosophical Logic, 27 (1998), pp. 217–274.

[2] O. Bernholtz, M. Vardi, and P. Wolper, *An automata-theoretic approach to branching-time model checking*, in Computer Aided Verification, Proc. 6th Int. Conference, Lecture Notes in Computer Science Nr. 818, Springer-Verlag, 1994, pp. 142–155.

[3] J. van Benthem, *Modal Logic and Classical Logic*, Bibliopolis, Napoli, 1983.

[4] J. van Benthem, *Dynamic bits and pieces*, ILLC research report, University of Amsterdam, 1997.

[5] J. van Benthem, *Process operation in extended dynamic logic*, in Proc. 13th IEEE Symp. on Logic in Computer Science, 1998, pp. 244–250.

[6] P. Blackburn, M. de Rijke, and Y. Venema, *Modal Logic*, Manuscript. Institute of Logic, Language and Information, University of Amsterdam, 1999. (A preliminary version is available at http://turing.wins.uva.nl/~mdr/Publications/modal-logic.html).

[7] E. Börger, E. Grädel, and Y. Gurevich, *The Classical Decision Problem*, Springer Verlag, 1997.

[8] R. Cleaveland and B. Steffen, *A linear-time model-checking algorithm for the alternation-free modal mu-calculus*, Formal Methods in System Design, 2 (1993), pp. 121–147.

[9] F. Donnini, M. Lenzerini, D. Nardi, and A. Schaerf, *Reasoning in description logics*, in Principles of Knowledge Representation, (G. Brewka, ed.), CSLI Publications, 1996, pp. 193–238.

[10] A. Emerson, *Temporal and modal logic*, in Handbook of Theoretical Computer Science, vol B., (J. van Leeuwen, ed.), Elsevier, 1990, pp. 995–1072.

[11] ——, *Model checking and the mu-calculus*, in Descriptive Complexity and Finite Models, N. Immerman and P. Kolaitis, eds., vol. 31 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, 1997, pp. 185–214.

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

[13] A. Emerson and C. Jutla, *The complexity of tree automata and logics of programs*, in Proc. 29th IEEE Symp. on Foundations of Computer Science, 1988, pp. 328–337.

[14] ——, *Tree automata, mu-calculus and determinacy*, in Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, pp. 368–377.

[15] E. Grädel, *On the restraining power of guards*, Journal of Symbolic Logic, 64 (1999), pp. 1719–1742.

[16] ——, *Decision procedures for guarded logics*, in CADE-16. Proceedings of 16th International Conference on Automated Deduction, 1999, pp. 31–51.

[17] E. Grädel, P. Kolaitis, and M. Vardi, *On the decision problem for two-variable first-order logic*, The Bulletin of Symbolic Logic, 3 (1997), pp. 53–69.

[18] E. Grädel and M. Otto, *On logics with two variables*, Theoretical Computer Science, 224 (1999), pp. 73–113.

[19] E. Grädel, M. Otto, and E. Rosen, *Undecidability results on two-variable logics*, in Proceedings of 14th Symposium on Theoretical Aspects of Computer Science STACS'97, Lecture Notes in Computer Science Nr. 1200, Springer, 1997, pp. 249–260.

[20] E. Grädel and E. Rosen, *On preservation theorems for two-variable logic*, Mathematical Logic Quarterly, 45 (1999), 315–325.

[21] E. Grädel and I. Walukiewicz, *Guarded fixed point logic*, in Proc. 14th IEEE Symp. on Logic in Computer Science, 1999, 45–54.

[22] Y. Gurevich, *On the classical decision problem*, Bulletin of the Europea Association for Theoretical Computer Science, 42 (1990), pp. 140–150.

[23] D. Janin and I. Walukiewicz, *On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic*, in Proceedings of 7th International Conference on Concurrency Theory CONCUR '96, no. 1119 in Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 263–277.

[24] D. Kozen, *A finite model theorem for the propositional μ-calculus*, Studia Logica, 47 (1983), pp. 233–241.

[25] ——, *Results on the propositional μ-calculus*, Theoretical Computer Science, 27 (1983), pp. 333–354.

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

[27] M. Mortimer, *On languages with two variables*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 21 (1975), pp. 135–140.

[28] S. Popcorn, *First Steps in Modal Logic*, Cambridge University Press, 1994.

[29] M. Rabin, *Decidability of second-order theories and automata on infinite trees*, Transactions of the AMS, 141 (1969), pp. 1–35.

[30] R. Streett and A. Emerson, *An automata theoretic decision procedure for the propositional mu-calculus*, Information and Computation, 81 (1989), pp. 249–264.

[31] M. Vardi, *Why is modal logic so robustly decidable?*, in Descriptive Complexity and Finite Models, N. Immerman and P. Kolaitis, eds., vol. 31 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, 1997, pp. 149–184.