6

Computability and Complexity

In this chapter we investigate the computability and complexity of normal modal logics. In particular, we examine the computability of *satisfiability problems* (given a modal formula ϕ and a class of models M, is it computable whether ϕ is M-satisfiable?) and *validity problems* (given a modal formula ϕ and a class of models M, is it computable whether ϕ is valid on M?). When the answer is 'yes', we probe further: how complex is the problem — in particular, what resources of time (that is, computation steps) or space (that is, memory) are needed to carry out the required computations? When the answer is 'no', we pose a similar question: how uncomputable is the problem? There are vast differences in the complexities of modal satisfiability problems: some are no worse than the satisfiability problem for propositional calculus, while others are highly undecidable.

This chapter has two main parts. The first, consisting of the five sections on the basic track, introduces the basic ideas and discusses modal (un-)decidability. Three techniques for proving decidability are discussed (finite models, interpretations in monadic second-order theories of trees, and quasi-models and mosaics) and undecidability is approached via tiling problems. In the second part, consisting of the last three sections of the chapter, we examine the complexity of some key modal satisfiability problems. These sections are on the advanced track, but the initial part of each of them should be accessible to all readers.

Basic ideas about computability and complexity are revised in the first section, and further background information can be found in Section C. Throughout the chapter we assume we are working with countable languages.

Chapter guide

Section 6.1: Computing Satisfiability (Basic track). In this section we introduce the key concepts assumed throughout the chapter: satisfiability and validity problems, and how to compute them on Turing machines.

Section 6.2: Decidability via Finite Models (Basic track). We discuss the use of finite models for proving decidability results. Three basic theorems are proved, and many of the logics discussed in Chapter 4 are shown to be decidable.

Section 6.3: Decidability via Interpretations (Basic track). Another way of proving modal decidability results is via interpretations in powerful decidable theories such as monadic second-order theories of trees. This technique is useful for showing the decidability of logics without the finite model property.

Section 6.4: Decidability via Quasi-models and Mosaics (Basic track). For logics lacking the finite model property it may also be possible to prove decidability results by computing with more abstract kinds of finite structure; quasi-models and mosaics are important examples of such structures.

Section 6.5: Undecidability via Tiling (Basic track). In this section we show just how easily undecidable — and even highly undecidable — modal logics can arise. We do so by introducing an important proof method: tiling arguments.

Section 6.6: NP (Advanced track). This section introduces the concept of NP algorithms, illustrates the modal content of this idea using some simple examples, and then proves Hemaspaandra's Theorem: every normal logic extending **S4.3** is NP-complete.

Section 6.7: PSPACE (Advanced track). The key complexity class for the basic modal language is PSPACE, the class of problems solvable in polynomial space. We give a PSPACE algorithm for the satisfiability problem for **K**, and prove Ladner's Theorem: every normal logic between **K** and **S4** is PSPACE-hard.

Section 6.8: EXPTIME (Advanced track). We show that the satisfiability problem for **PDL** is EXPTIME-complete. EXPTIME-hardness is shown by reduction from a tiling problem, and the EXPTIME algorithm introduces an important technique called elimination of Hintikka sets.

6.1 Computing Satisfiability

The work of this chapter revolves around satisfiability and validity problems. Here is an abstract formulation.

Definition 6.1 (Satisfiability and Validity Problems) Let τ be a modal similarity type, ϕ be a τ-formula and M a class of τ-models. The M-*satisfiability problem* is to determine whether or not ϕ is satisfiable in some model in M. The M-*validity problem* is to determine whether or not ϕ is true in all models in M; that is, whether or not $\mathsf{M} \Vdash \phi$. (We call this the validity problem because we are mostly interested in cases where M is the class of all models over some class of frames.) The M-validity and M-satisfiability problem are each other's *duals*. ⊣

In fact, as far as discussions of computability (or non-computability) are concerned, we are free to talk in terms of either satisfiability or validity problems.

Lemma 6.2 *Let τ be a modal similarity type, and suppose that M is a class of τ-models. Then there is an algorithm for solving the M-satisfiability problem iff there is an algorithm for solving the M-validity problem.*

Proof. As $\neg\phi$ is *not* satisfiable in M iff $\mathsf{M} \Vdash \phi$, given an algorithm for M-satisfiability, we can test for the validity of ϕ by giving it the input $\neg\phi$. In a similar fashion, an algorithm for M-validity can be used to test for M-satisfiability. ⊣

This argument does not give us any interesting information about the relative *complexity* of dual satisfiability and validity problems; and indeed, they may well be different.

How do the themes of this chapter relate to the *normal modal logics* introduced in Section 1.6 and discussed in Chapters 4 and 5? Clearly we should investigate the following two problems.

Definition 6.3 Let τ be a modal similarity type, Λ be a normal modal logic in a language for τ, and ϕ a τ-formula. The problem of determining whether or not ϕ is Λ-consistent is called the *Λ-consistency problem*, and the problem of determining whether or not $\Lambda \vdash \phi$ is called the *Λ-provability problem*. ⊣

Note that Λ-consistency and Λ-provability problems are satisfiability and validity problems in disguise. In particular, if Λ is a normal modal logic, and M is any class of models such that $\Lambda = \Lambda_{\mathsf{M}}$, then the Λ-consistency problem is the M-satisfiability problem, and the Λ-provability problem is the M-validity problem. As every normal modal logic is determined by at least one class of models (namely, the singleton class containing its canonical model; see Theorem 4.22), we are free to think of consistency and provability problems in terms of satisfiability and validity problems. We do so in this chapter, and to emphasize this we usually call the Λ-consistency problem the *Λ-satisfiability problem*, and the Λ-provability problem the *Λ-validity problem*.

Our discussion so far has given an *abstract* account of the problems we will explore, and most of our results will be stated, proved, and discussed at this level. But what does it mean to have an algorithm for solving (say) a validity problem? And what does it mean to talk about the complexity of (say) a satisfiability problem? After all, computation is the finitary manipulation of finite structures — but both formulas and models are abstract set-theoretical objects. To show that our abstract account really makes sense, we need to choose a well-understood method of computation and show that formulas and models can be *represented* in a way that is suited to our method.

We have chosen *Turing machines* (Section C) as our fundamental model of computation. The most relevant fact about Turing machines for our purposes is that they compute by manipulating finite strings of symbols; hence we need to represent models and formulas as symbol strings. As far as mere computability is concerned, the key demand is that these symbol string representations be *finite*. For complexity analyses more is required: representations must also be *efficient*. Let's discuss these requirements.

Clearly modal formulas can be represented as finite strings over a finite set of symbols: proposition letters can be represented by a single symbol (say, p) followed by (the representation of) a number. Thus, instead of working with an infinite collection of primitive symbols we could work with (say) $p1$, $p10$, $p11$, $p100$ and so on, where the numeric tail is represented in binary. Fine — but what about models? Models are set-theoretic entities of the form (W, R, V), and each component may be infinite. However, the difficulty is more apparent than real. For a start, when evaluating a formula ϕ in some model, the only relevant information in the valuation is the assignments made to propositional letters actually occurring in ϕ (see Exercise 1.3.1). Thus, instead of working with V, we can work with the finite valuation V' which is defined on the (finite) language consisting of exactly the proposition letters in ϕ, and which agrees with V on these letters. Secondly, much of our work will revolve around models based on *finite* frames (or more generally, the frames of *finite character* defined below).

We already know quite a lot about finite models and their logics. For a start, in Section 2.3 we introduced two techniques for building finite models (selection and filtration) and defined the finite model property for the basic modal language. In Section 3.4 we introduced the finite frame property (again, for the basic modal language) and proved Theorem 3.28: a normal modal logic has the finite frame property iff it has the finite model property. Since then we have learned what a normal modal logic in a language of arbitrary similarity type is (Definition 4.13), so let's now define the finite frame property and the finite model property for modal languages of arbitrary similarity type, and generalize Theorem 3.28.

Definition 6.4 Let τ be a modal similarity type. A frame of type τ has *finite character* if it contains finitely many states, and finitely many non-empty relations. If Λ is a normal modal logic in a language for τ, and F is a class of τ-frames of finite character, and $\Lambda = \Lambda_{\mathsf{F}}$, then Λ is said to have the *finite frame property (f.f.p.)* with respect to F. If $\Lambda = \Lambda_{\mathsf{F}}$ for some class of τ-frames F of finite character, then Λ has the finite frame property.

A class of τ-models M is *finitely based* if every model in M is based on a τ-

frame of finite character. If Λ is a normal modal logic in a language for τ, and M is a class of finitely based τ-models, and $\Lambda = \Lambda_{\mathsf{M}}$, then Λ has the *finite model property (f.m.p.)* with respect to M. If $\Lambda = \Lambda_{\mathsf{M}}$ for some class of of finitely based τ-models M, then Λ has the finite model property. $\dashv$

A few remarks may be helpful. First, the concept of finite character is a natural way of coping with similarity types containing infinitely many relations. Second, note that the way the finite frame property is defined here (where we simply insist that $\Lambda = \Lambda_{\mathsf{F}}$) is somewhat simpler than that used in Definition 4.13 (where we insisted that $\mathsf{F} \Vdash \Lambda$, and for every formula ϕ such that $\phi \notin \Lambda$ there is some $\mathfrak{F} \in \mathsf{F}$ such that ϕ is falsifiable on $\mathfrak{F}$). It is easy to see that these definitions are equivalent Finally, a class of frames of finite character (or indeed, a class of finite frames) may well be a proper class. Nonetheless, up to isomorphism, there are only denumerably many frames in any such class; hence, if Λ has the finite frame property, it has the finite frame property with respect to a denumerably infinite *set* of frames, and we take this for granted without further comment throughout the chapter.

Given this definition, it is straightforward to generalize Theorem 3.28.

Theorem 6.5 *Let τ be a modal similarity type. Any normal modal logic in a language for τ has the finite model property iff it has the finite frame property.*

Proof. This is a matter of verifying that the proof of Theorem 3.28 extends to arbitrary similarity types; see Exercise 6.1.1. $\dashv$

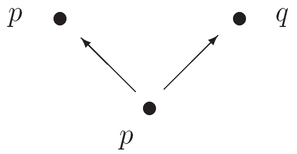
There are many ways to represent a frame of finite character, together with a valuation V' defined on finitely many proposition letters, as a finite symbol string. While any such finitization is sufficient for discussions of computability, we need to exercise more care when it comes to complexity. Complexity theory measures the difficulty of problems in terms of the resources required to solve them — and these are measured as a function of the size of the input. A highly inefficient representation of the input can render such resource measures vacuous, so we must be careful not to smuggle in sources of inefficiency. For the complexity classes we will be dealing with, this is pretty much a matter of common sense, but the following point should be made explicit: we must *not* represent the numeric subscripts on propositional variables and states in unary notation.

The point is this. Even binary representations (which are longer than the more familiar decimal representations) are exponentially more compact than unary ones. For example, the representation of the number 64 in unary is a string of 64 consecutive ones, whereas its representation in binary is 1000000. If we represent our subscripts in unary, we are using a *highly* inefficient representation of the problem.

For this reason we will regard modal formulas (for the basic modal language) as strings over the alphabet $\{p, 0, 1, (,), \wedge, \neg, \diamond\}$, and proposition letters will be represented by strings consisting of p followed by the *binary* representation of a number (without leading zeroes). Similarly, we will regard models as strings over the alphabet $\{w, p, 0, 1, ;, \langle, \rangle\}$. A state in a model will be represented by w followed by the binary representation of a number (without leading zeroes), and the representation of proposition letters (which we need to encode the valuation) will be as just described. A string representing a model will have the following form:

$$\begin{aligned}&\big\langle \langle w_1; \ldots ; w_n \rangle;\\ &\ \ \langle \langle w_i; w_j \rangle; \ldots ; \langle w_k; w_l \rangle \rangle;\\ &\ \ \langle \langle p_x; \langle w_r; \ldots ; w_s \rangle \rangle; \ldots ; \langle p_y; \langle w_t; \ldots ; w_u \rangle \rangle \big\rangle,\end{aligned}$$

where $1 \leq i, j, k, l, r, s, t, u \leq n$. Such triples represent models in the obvious way: the first component gives the states, the second the relation, and the third the valuation. The subscripted w's and p's are metavariables over our representations of states and proposition letters, respectively. We assume that our representations of models contain no repetitions in any of the three components, and that they satisfy obvious well-formedness conditions (in particular, the third component represents a *function*, thus we cannot have the same representation p_y appearing as the first item in different tuples). Here is a simple example (though to keep things readable we have represented the numbers in decimal):



A model

$$\begin{aligned}&\big\langle \langle w_1; w_2; w_3 \rangle;\\ &\ \ \langle \langle w_1; w_2 \rangle; \langle w_1; w_3 \rangle \rangle;\\ &\ \ \langle \langle p_1; \langle w_1; w_2 \rangle \rangle; \langle p_2; \langle w_3 \rangle \rangle \rangle \big\rangle\end{aligned}$$

Its representation

Such representations open the door to all the standard concepts of computability theory and computational complexity. For a start, it now makes sense to describe sets of formulas (including normal modal logics), sets of models, and sets of frames as being *recursively enumerable (r.e.)*, or as being *recursive*. Saying that a set is r.e. means that it is possible to write a Turing machine that will successively output all and only its elements. Saying that a set is recursive means that it is possible to write a Turing machine which, when given any input, will perform a *finite* number of computation steps, halt, and then correctly tell us whether the input represents a member of the set or not. (In short, recursive sets are those for which we can decide membership using a terminating computation.)

Furthermore, it is clearly possible to program a Turing machine so that when it is presented with (the representations of) a formula, a model, and a point, it will evaluate (the representation of) the formula in (the representation of) the model at (the representation of) the point. Admittedly it would be rather painful to write out such a Turing machine in detail — but it is straightforward to write a program to carry out this task in most high-level programming languages; hence, by Church's Thesis (see Section C), it is possible to write a Turing machine to do the job as well. Thus it makes perfectly good sense to talk about writing Turing machines which test for the satisfiability or validity of a formula on a class of finitely based models and to inquire about the complexity of such problems.

Apart from asking the reader to generalize the above representation schema to cover modal languages of arbitrary similarity type (see Exercise 6.1.2) we will not discuss the issue of representation further. In most of what follows we talk as if the abstract definition of satisfiability and validity problems given earlier was the focus of our computational investigations. For example, we will often call $|\phi|$ the size of the input formula; strictly speaking, it is the size of its representation. Nor do we mention Turing machines very often. The results of this chapter rest on the fact that there is an efficient representation which enables us to compute satisfiability and validity problems; for many purposes we can ignore the details.

Exercises for Section 6.1

6.1.1 Prove Theorem 6.5. That is, show that for any modal similarity type τ, any normal modal logic in a language for τ has the finite model property if and only if it has the finite frame property. This is simply a matter of verifying that the proof of Theorem 3.28 extends to arbitrary similarity types — but note that there will be a gap in your proof if you haven't yet proved the Filtration Theorem for modal languages of arbitrary similarity type.

6.1.2 Modify the representation schema for models given above so that it can represent any finitely based model of any modal similarity type.

6.1.3 Show that if Λ is the normal modal logic generated by an r.e. set of formulas, then Λ itself is an r.e. set. (The reader unfamiliar with this type of proof may find it useful to look at the proof of Lemma 6.12 below.)

6.2 Decidability via Finite Models

Call a normal modal logic Λ *decidable* if the Λ-satisfiability (or equivalently: Λ-validity) problem is decidable, and *undecidable* if it is not. How should we establish decidability results? A lot depends on our 'access' to the logic. For example, we may know Λ purely semantically: it is given as the logic of some class of frames of interest. However, we may also have a syntactic handle on Λ; in particular, we may know that it is the logic generated by some set of axioms. Whether Λ is semantically or syntactically specified, establishing that it has the finite model property is a useful first step towards proving decidability, for if we can prove this, two plausible strategies for establishing decidability suggest themselves, as we will now explain.

- ***Decidability for semantically specified logics: informal argument.*** Suppose we only have a semantic specification of Λ, but that we have been able to prove that Λ possesses a strong form of the finite model property: not only does Λ have the f.m.p. with respect to some set of models, but for any formula ϕ there is a computable function f such that $f(|\phi|)$ is an upper bound on the size of these models needed to satisfy ϕ. Write a Turing machine that takes ϕ as input, generates all the finite models belonging to this set up to size $f(|\phi|)$, and tests for the satisfiability of ϕ on these models. Because ϕ is Λ-satisfiable iff it is satisfied in a Λ-model of size at most $f(|\phi|)$, and because the machine systematically examines all these models, our machine decides Λ-satisfiability.
- ***Decidability for syntactically specified logics: informal argument.*** Suppose Λ is given axiomatically, and we have been able to show that Λ has the f.m.p. with respect to some set of models M. First, construct a Turing machine that makes use of the axiomatization to recursively enumerate the Λ-validities. Second, construct a Turing machine that recursively enumerates all the finite models in M. Given two such machines we can effectively test the Λ-validity of any formula ϕ: if ϕ is valid it will eventually be generated by the first machine; if it is not, we will eventually be able to falsify it on a model generated by the second. One of the machines must eventually settle ϕ's fate, and thus decide Λ-validity.

Such arguments underly most applications of the finite model property to decidability. We have deliberately phrased both arguments rather loosely; the fundamental goal of this section is to explore the underlying ideas more carefully, and formulate them rigorously. Our investigation will yield three main theorems. The first is a precise formulation of the argument for semantically specified logics. The second and third are distinct reformulations of the argument for syntactically specified logics. We will consider a number of applications of these theorems, and will put both of the methods introduced in Section 2.3 for constructing finite models (namely *filtration* and *selection*) to work.

Let us begin by scrutinizing the first of the above arguments. This revolves around a strong form of the finite model property.

Definition 6.6 (Strong Finite Model Property) Let Λ be a normal modal logic, M a set of finitely based models such that $\Lambda = \Lambda_{\mathsf{M}}$, and f a function mapping natural numbers to natural numbers. Λ has the $f(n)$-*size model property* with respect to M if every Λ-consistent formula ϕ is satisfiable in a model in M containing at most $f(|\phi|)$ states.

Λ has the *strong finite model property* with respect to M if there is a *computable* function f such that Λ has the $f(n)$-size model property with respect to M. Λ has the *polysize model property* with respect to M if there is a *polynomial* p such that Λ has the $p(n)$-size model property with respect to M.

Λ has the $f(n)$-size model property (respectively, strong finite model property, polysize model property) if there is a set of finitely based models M such that $\Lambda = \Lambda_{\mathsf{M}}$ and Λ has the $f(n)$-size model property (respectively, strong finite model property, polysize model property) with respect to M. $\dashv$

If a logic Λ has the polysize model property, any Λ-satisfiable formula is satisfiable not just on a finite model, but a genuinely *small* model. Even this very strong form of the f.m.p does *not* guarantee decidability: as the reader is asked to prove in Exercise 6.2.4, there are uncountably many normal modal logics which possess the polysize model property but have *undecidable* satisfiability problems.

In view of this result, the first informal argument sketch is clearly inadequate — but where does its deficiency lie? It makes the following (false) assumption: that for any set of models, and any natural number n, it is possible to generate all and only the models in M of size at most n. This assumption is warranted only if M is a recursive set (that is, only if a Turing machine can decide exactly which finite models belong to M). But this is the only shortcoming of the informal argument.

Theorem 6.7 *If Λ is a normal modal logic that has the strong finite model property with respect to a recursive set of models M, then Λ is decidable.*

Proof. First, observe that for any natural number n it is possible to generate all distinct (representations of) models in M that have size at most n: we need simply write a machine that generates *all* distinct (representations of) models that have size at most n, tests each model (representation) as it is generated to see whether it belongs to M (this is the key point: we can effectively test for membership in M precisely because M is a recursive set) and then outputs exactly those models (representations) which do belong to M. (From now on we drop all mention of representations, and will speak simply of 'generating all models' or 'generating all models up to size n', and so on.)

So, given ϕ, we use this machine to generate all models of the appropriate set up to size $f(\phi)$, and test whether ϕ is satisfiable on any of the models it produces. If ϕ is satisfiable on at least one of them, it is Λ-satisfiable; if not, it is not Λ-satisfiable, for Λ has the strong f.m.p. with respect to M. $\dashv$

Theorem 6.7 is an important result. If we are to apply it, how do we establish that a logic has the strong finite model property? Unfortunately, no fully general answer to this question is known — nonetheless, both filtration and selection can be useful. We start by illustrating the utility of filtrations.

Corollary 6.8 **K**, **T**, **KB**, **K4**, **S4**, **S5**, $\mathbf{K}_t$, $\mathbf{K}_t4.3$ *and* $\mathbf{K}_t\mathbf{Q}$ *are decidable.*

Proof. First, all these logics have the f.m.p. with respect to the expected sets of models; for example, **K4** has the f.m.p. with respect to the set of finite transitive models, and $\mathbf{K}_t\mathbf{Q}$ has the f.m.p with respect to the finite dense unbounded weak total orders (that is, the finite DUWTO frames; see Theorem 4.41). The easiest way to prove this is to use filtrations. In Section 2.3 we defined filtrations for both the basic modal language and the basic temporal language. Given a model $\mathfrak{M}$ that satisfies a formula ϕ at some state, by filtrating $\mathfrak{M}$ through the set of all ϕ's subformulas we obtain a *finite* model $\mathfrak{M}^f$ that satisfies ϕ. Of course, we need to be careful that $\mathfrak{M}^f$ has all the right properties; for example, if $\mathfrak{M}$ was a **K4**-model, we want $\mathfrak{M}^f$ to be a **K4**-model as well. By and large this is straightforward, though the reader will need to think a little about how to handle density; see Exercise 6.2.1.

Such filtration arguments actually establish the *strong* f.m.p. for these logics. If we form $\mathfrak{M}^f$ by filtrating $\mathfrak{M}$ through the subformulas of ϕ, then $\mathfrak{M}^f$ has at most $2^{|\phi|}$ nodes, thus we have a computable (though, unfortunately, exponential) upper bound on the size of satisfying models for all these logics; see Section 2.3.

It remains to check that the relevant sets of finite models are recursive. Checking for membership in these sets boils down to checking that the models possess (various combinations of) such properties as reflexivity, transitivity, trichotomy, and so on. It is clearly possible to devise algorithms to test for the relevant properties, hence (by Church's thesis) we can program a Turing machine to do so. Thus Theorem 6.7 applies, and all these logics are decidable. $\dashv$

Filtration is a widely used technique for showing that logics have the strong finite model property, but it has limitations. Suppose we are working with a modal language containing n unary modal operators ($n > 0$) and no others. Let F_1^n be the set of frames for this language such that for each $\mathfrak{F} \in \mathsf{F}_1^n$, the relation corresponding to each modality is a partial function, let M_1^n be the set of models built over F_1^n, and let $\mathbf{K}_n\mathbf{Alt}_1$ be its logic. Now, $\mathbf{K}_n\mathbf{Alt}_1$ has the strong finite model property, but there is no obvious way of using filtrations to show this; see Exercise 6.2.3.

However — at least in the present case — it is straightforward to use *selection*, the other method of building finite models discussed in Section 2.3, to establish the strong finite model property.

Corollary 6.9 $\mathbf{K}_n\mathbf{Alt}_1$ *is decidable.*

Proof. We argue as follows. Suppose $\mathfrak{M}$ is in M_1^n and $\mathfrak{M}, w \Vdash \phi$. Let $\mathfrak{M}'$ be the model that is identical to $\mathfrak{M}$ save possibly that any relations in $\mathfrak{M}'$ *not* corresponding to modal operators in ϕ are empty. Clearly $\mathfrak{M}'$ is also in M_1^n and $\mathfrak{M}', w \Vdash \phi$. Let m be the degree of ϕ (that is, the maximal depth of nested modalities; see Definition 2.28). Let $\mathfrak{M}''$ be the submodel of $\mathfrak{M}'$ formed by selecting all and only those nodes reachable from w in m or fewer steps. Clearly $\mathfrak{M}''$ is in M_1^n and $\mathfrak{M}'', w \Vdash \phi$. Moreover, because each relation is a partial function, $\mathfrak{M}''$ has only finitely many nodes: indeed, it can contain at most $t^m + 1$ nodes, where t is the number of distinct types of modality that occur in ϕ. Hence $\mathbf{K}_n\mathbf{Alt}_1$ has the strong finite model property with respect to M_1^n.

It is clear that the set of finitely based M_1^n models is recursive, for testing whether a finite model $\mathfrak{M}$ belongs to it essentially boils down to checking that each of $\mathfrak{M}$'s (finitely many non-empty) transition relations is a partial function. Decidability follows by Theorem 6.7. $\dashv$

Selection is not as general a method as filtration — but it can be useful, especially when working with non-transitive models. As we will see when we discuss NP-completeness, selection is a natural way of turning a finite model (perhaps produced via a filtration) into a truly small (that is, polysize) model.

Theorem 6.7, together with such methods as filtration and selection, can be a useful tool for establishing modal decidability results, for it does not require us to have an axiomatization. Very often we do have an axiomatization at our disposal, and it is natural to ask whether (and how) we can make use of it to help establish decidability. This is what the second informal argument attempts to do. The key idea it embodies is the following: if a logic is both axiomatizable and has the finite model property with respect to some (recursively enumerable) set of models M, then we should be able to prove decidability. This is an important idea that can be developed in two different ways, depending on the kind of axiomatization we have, and what we know about the computational properties of M.

When we discussed completeness in Chapter 4, we viewed axiomatizations very abstractly: we simply said that if Λ was a normal modal logic, Σ a set of modal formulas, and $\mathbf{K\Sigma}$ (the smallest normal logic generated by Σ) equaled Λ, then Σ was an axiomatization of Λ. To give computational content to the phrase 'generated by' we need to impose restrictions on Σ, for under the definition just given every normal logic Λ generates itself. This is too abstract to be useful here, so we will introduce various notions of *axiomatizability* that offer more computational leverage.

Definition 6.10 A logic Λ is *finitely axiomatizable* if it has a *finite* axiomatization Σ; it is *recursively axiomatizable* if it has a *recursive* axiomatization Σ; and it is *axiomatizable* if it has a *recursively enumerable* axiomatization Σ. $\dashv$

Although it won't play a major role in what follows, there is a neat result called Craig's Lemma that readers should know: *every axiomatizable logic is recursively axiomatizable*. So the following lemma is essentially Craig's Lemma for modal logic:

Lemma 6.11 *If Λ is axiomatizable, then Λ is recursively enumerable.*

So, given a computationally reasonable notion of axiomatizability, the idea of using axiomatizations to generate validities is correct. But how do we use this fact to turn the informal argument into a theorem? Here's the most obvious way: demand that M be an r.e. set. As the following lemma shows, this ensures that we can recursively enumerate the formulas that are *not* valid on M.

Lemma 6.12 *If* M *is a recursively enumerable set of finite models, then the set of formulas falsifiable in* M *is recursively enumerable.*

Proof. As M is an r.e. set, we can construct a machine *M1* to generate all its elements, and clearly we can construct a machine *M2* that generates all the formulas. So, construct a machine *M3* that operates as follows: it calls on *M1* to generate a model, and on *M2* to generate a formula, and then stores both the model and the formula. It then tests all stored formulas on all stored models (*M3* is not going to win any prizes for efficiency) and outputs any of the stored formulas it can falsify on some stored model. *At any stage there are only finitely many stored formulas and models, hence this testing process terminates.* When the testing process is finished, *M3* calls on *M1* and *M2* once more to generate another model and formula, stores them, performs another round of testing, and so on *ad infinitum*.

Suppose ϕ is falsifiable on some model $\mathfrak{M}$ in M. At some finite stage both ϕ and $\mathfrak{M}$ will be stored by *M3*, hence ϕ will eventually be tested on $\mathfrak{M}$, falsified, and returned as output. This means that the set of formulas falsifiable on $\mathfrak{M}$ is recursively enumerable. $\dashv$

Theorem 6.13 *If* Λ *is an axiomatizable normal modal logic that has the finite model property with respect to an r.e. set of models* M*, then* Λ *is decidable.*

Proof. Λ is r.e. by Lemma 6.11. But the set of formulas *not* in Λ is also r.e. for $\Lambda = \Lambda_{\mathsf{M}}$ and the set of formulas that are *not* M-valid is r.e. by the previous lemma. Any formula ϕ must eventually turn up on one of these enumerations, hence Λ is decidable. $\dashv$

As an application, we will show that the minimal propositional dynamic logic is decidable.

Corollary 6.14 **PDL** *is decidable.*

Proof. By Theorem 4.91, **PDL** is complete with respect to the set of all regular PDL-models. The axioms of **PDL** clearly form a *recursive* set, so trivially they form a recursively enumerable set, thus to be able to apply the Theorem 6.13 it only remains to show that **PDL** has the finite model property with respect to an r.e. set of models.

This follows easily from our completeness proof for **PDL**. Recall that we proved completeness by constructing, for any consistent formula ϕ, a finite model $\mathfrak{P}$ that satisfied ϕ. This gives us what we want, modulo the following glitch: although

$\mathfrak{P}$ contains only finitely many nodes, it may contain infinitely many non-empty relations, thus it may not be of finite character and thus (strictly speaking) our completeness proof does not establish that **PDL** has the finite model property. This is a triviality: for any formula ϕ, only finitely many of the relations on $\mathfrak{P}$ are relevant to the satisfiability of ϕ, namely those that actually occur in ϕ. Let $\mathcal{R}_\phi$ be the smallest set that contains all the relations in $\mathfrak{P}$ corresponding to modalities in ϕ and is downward closed under the usual relation constructors (that is, if $R_{\pi;\pi'} \in \mathcal{R}_\phi$ then so are R_π and $R_{\pi'}$, and analogously for relations defined by union and transitive closure). Note that $\mathcal{R}_\phi$ is finite. Let $\mathfrak{P}'$ be the model that is identical to $\mathfrak{P}$ save that all the relations *not* in $\mathcal{R}_\phi$ are empty; we call $\mathfrak{P}'$ a *reduced model*. Clearly $\mathfrak{P}'$ is a finitely based model that satisfies ϕ. This shows that **PDL** has the finite model property.

The set of reduced models is a recursive set, since checking that a finite model is a reduced model boils down to showing that the relations corresponding to non-basic modalities really are generated out of simpler relations via composition, union, or transitive closure, and this is obviously something we can write a program to do. Hence, the relevant models are recursively enumerable, thus the conditions of Theorem 6.13 are satisfied, and **PDL** is decidable. $\dashv$

We can also show that **PDL** is decidable by appealing to Theorem 6.7. As we have just seen, our completeness proof for **PDL** gives us the finite model property for **PDL** — but in fact it even gives us the *strong* finite model property. To see this, recall that for any consistent ϕ, we constructed $\mathfrak{P}$ out of *atoms*, that is, maximal consistent subsets of the Fisher-Ladner closure of $\{\phi\}$. As there are at most $2^{c|\phi|}$ such atoms for some constant c, we have a computable upper bound on the size of the models needed to satisfy ϕ. We noted in the proof of Corollary 6.14 that the relevant finite models (the reduced models) form a recursive set, hence we have established everything we need to apply Theorem 6.7.

Theorem 6.13 is a fundamental one and is useful in practice. It does not make use of axiomatizations in a particularly interesting way: it uses them merely to enumerate validities. To apply the theorem we need to know that the set of relevant finite models is recursively enumerable. We often have much stronger syntactic information at our disposal: we may know that a logic is *finitely* axiomatizable. Our next theorem is based on the following observation: if a logic with the f.m.p. is *finitely* axiomatizable, we can use the axiomatization not only to recursively enumerate the validities, but to help us enumerate the non-validities as well.

Theorem 6.15 *If Λ is a finitely axiomatizable normal modal logic with the finite model property, then Λ is decidable.*

Proof. As in the proof of Theorem 6.13 we can use the axiomatization to recursively enumerate Λ, so if we can show that the set of formulas *not* in Λ is also r.e. we will have proved the theorem.

By Theorem 6.5, if Λ has the finite *model* property it also has the finite *frame* property, thus there is some set of finite frames F such that $\Lambda = \Lambda_{\mathsf{F}}$. Hence, if $\phi \notin \Lambda$, ϕ is falsifiable in some model based on a frame in F. Obviously all such frames must validate every axiom of Λ, hence if $\phi \notin \Lambda$, ϕ is falsifiable in some model based on a frame that validates the Λ axioms. Now for the crucial observation: we can write a machine M which decides whether or not a finite frame validates the Λ axioms, for as Λ has only finitely many axioms, each frame can be checked in finitely many steps. With the help of M, we can recursively enumerate the formulas falsifiable in some F-based model, but these are just the formulas which do not belong to Λ. It follows that Λ is decidable. $\dashv$

Can Theorem 6.15 be strengthened by replacing its demand for a finite axiomatization with a demand for a *recursive* axiomatization? No — in Exercise 6.2.5 we give an example of an *undecidable* recursively axiomatizable logic $\mathbf{KU}_X$ with the finite model property; the result hinges on Craig's Lemma.

Theorem 6.15 has many applications, for many common modal and tense logics have the f.m.p. and are finitely axiomatizable. For example, Theorem 6.15 yields another proof that **K**, **T**, **KB**, **K4**, **S4**, **S5**, $\mathbf{K}_t$, $\mathbf{K}_t\mathbf{4.3}$, and $\mathbf{K}_t\mathbf{Q}$ are decidable, for all these logics were shown to be finitely axiomatizable in Chapter 4, and we saw above that they all have the (strong) finite model property. However, a more interesting application follows from our work on logics extending **S4.3** in Section 4.9.

Corollary 6.16 *Every normal logic extending* **S4.3** *is decidable.*

Proof. By Bull's Theorem (Theorem 4.96) every normal logic extending **S4.3** has the finite model property, and by Theorem 4.101 every normal logic extending **S4.3** is finitely axiomatizable. Hence the result is an immediate corollary of Theorem 6.15. $\dashv$

Corollary 6.16 completes the main discussion of the section. To summarize what we have learned so far, in Theorems 6.7, 6.13, and 6.15 we have results that pin down three important situations in which the finite model property implies decidability — and indeed, most modal decidability results make use of one of these three theorems.

Exercises for Section 6.2

6.2.1 Provide full proof details for Corollary 6.8. Pay particular attention to showing that $\mathbf{K}_t\mathbf{Q}$ has the f.m.p. with respect to the finite DUWTO-frames (see Theorem 4.41). Filtrations generally don't preserve density, so how do we know that this filtration is dense? (Hint: trichotomy.)

6.2.2 Show that if Λ is a finitely axiomatizable normal modal logic with the finite model property, then Λ has the finite frame property with respect to a recursive set of frames.

6.2.3 . In this exercise we ask you to show that there is method of filtrating a partial function that guarantees that the resulting relation is again a partial function.

Consider the model $\mathfrak{M} = (\mathbb{N}, S, V)$ where S is the successor relation on the set $\mathbb{N}$ of natural numbers, and V makes the proposition letter p true at precisely the even numbers. Let Σ be the set $\{\Diamond\neg p, \Diamond p, \neg p, p\}$. Prove that no filtration of $\mathfrak{M}$ through Σ is based on a frame in which S^f is a partial function.

6.2.4 In this exercise we ask the reader to prove that there are uncountably many undecidable normal modal logics with the polysize model property.

Let F_{suc} be the set of all finite frames (W, R) such that $W = \{0, \ldots, k\}$ (for some $k \in \omega$) and for all $0 \leq n < m \leq k$, Rnm iff $m = n + 1$. (Note that this definition permits reflexive points. Indeed, any frame in this set is uniquely determined by its size and which points, if any, are reflexive.) Then, for each $j \in \omega$ define F_j to be the set containing: (1) all the irreflexive frames in F_{suc}; (2) all the frames in F_{suc} whose last point is reflexive; and (3) the (unique) F_{suc} frame containing $j + 1$ nodes such that 0 is the only reflexive point; call this frame $\mathfrak{F}_j$. Now define, for any non-empty $I \subseteq \omega$, F_I as the set $\bigcup_{i \in I} \mathsf{F}_i$; let Λ_I be its logic.

Define ϕ_j to be the formula $p \wedge \Diamond p \wedge \Diamond(\neg p \wedge \Diamond^{j-1}\Box\bot)$.

(a) Prove that ϕ_j is satisfiable in F_i iff $i = j$.
(b) Prove that if ϕ is F_i-satisfiable, then it is satisfiable on a frame in F_i that contains at most $m + 2$ points, where m is the number of modalities in ϕ.
(c) Prove that if I and J are distinct (non-empty) subsets of ω then there is a formula that is satisfiable in F_I but not in F_J.
(d) Prove that each Λ_I has the polysize model property.
(e) Prove that there can only be countably many decidable logics. (This step is actually the easiest one: after all, how many distinct Turing machines can there be?)
(f) Conclude that there are uncountably many undecidable normal modal logics with the polysize model property.

6.2.5 Let X be an r.e. subset of the natural numbers that is *not* recursive; assume that $0 \in X$ but $1 \notin X$. Then $\mathbf{KU}_X$ is the smallest normal modal logic containing the following formulas:

(U1) $\Diamond(\Diamond p \wedge \Diamond q) \to \Diamond\Diamond(p \wedge q)$
(U2) $\Diamond(p \wedge \Box\bot) \wedge \Diamond(q \wedge \Box\bot) \to \Diamond(p \wedge q)$
(U3) $\Diamond(p \wedge \Diamond\top) \wedge \Diamond(q \wedge \Diamond\top) \to \Diamond(p \wedge q)$
(U4)$_k$ $(\Diamond\Box\bot \wedge \Diamond\Diamond\top) \to \Box^k\Diamond\top$, where $k \in X$.

Note that by Craig's Lemma $\mathbf{KU}_X$ has a *recursive* axiomatization.

(a) Use Sahlqvist's Correspondence and Completeness Theorem to find a first order definable class U of frames for which $\mathbf{KU}_X$ is sound and complete.
(b) Prove that $\mathbf{KU}_X$ has the finite model property.
(c) Show that $\mathbf{KU}_X$ is undecidable.
(Hint: prove that any formula $\mathsf{U4}_j$ with j *not* in X, is not satisfiable in U.)

6.3 Decidability via Interpretations

For all its usefulness, decidability via finite models has a number of limitations. One is absolute: as we will shortly see, there are decidable logics that lack the finite model property. Another is practical: it may be difficult to establish the finite model property, for although filtration or selection work in many cases, no universal approach is known. Thus we need to become familiar with other techniques for establishing decidability, and in this section we introduce an important one: *decidability via interpretations*, and in particular, *interpretations in* SnS.

A general strategy for proving a problem decidable is to effectively reduce it to a problem already known to be decidable. But there are many decidable problems; which of them can help us prove modal decidability results? Ideally, we would like to find a decidable problem, or class of problems, to which modal satisfiability problems can be reduced in a reasonably natural manner. Moreover, we would like the approach to be as general as possible: not only should a large number of modal satisfiability problems be so reducible, but the required reductions should be reasonably uniform.

A suitable group of problems is the satisfiability problem for SnS (where $n \in \omega$ or $n = \omega$), the monadic second-order theory of trees of infinite depth, where each node has n successors. Because these problems are themselves satisfiability problems — and indeed, satisfiability problems for monadic second-order languages, the kinds of language used in correspondence theory — it can be relatively straightforward to reduce modal satisfiability to SnS satisfiability. Moreover, the various reductions share certain core ideas; for example, analogs of the standard translation play a useful role. The method can also be used for strong modal languages, such as languages containing the until operator U, see Exercise 2.2.4.

In this section we introduce the reader to such reductions (or better, for reasons which will become clear, *interpretations*). We first introduce the theories SnS, note some examples of their expressivity, and state the crucial decidability results on which subsequent work depends. We then illustrate the method of interpretations with two examples. First, we prove that **KvB**, a logic lacking the finite model property, is decidable. As **KvB** is characterized by a *single* structure (namely, a certain general frame) this example gives us a relatively straightforward introduction to the method. We then show how the decidability of **S4** can be proved via interpretation. The result itself is rather unexciting — we already know that **S4** has the finite model property and is decidable (see Corollary 6.8) — but the proof is important and instructive. **S4** is most naturally characterized as the logic of transitive and reflexive frames, but this is a characterization in terms of an uncountable class of structures. How can this characterization be ‘interpreted’ in SnS? In fact, it can be done rather naturally, and the ideas involved open the doors to a wide range of further decidability results.

Let us set about defining SnS. If $\mathcal{A}$ is some fixed set (our alphabet), then $\mathcal{A}^*$ is the set of all finite sequences of elements of $\mathcal{A}$, including the null-sequence λ. We introduce the following apparatus:

(i) Define an ordering $\leq$ on $\mathcal{A}^*$ by $x \leq y$ if $y = xz$ for some $z \in \mathcal{A}^*$. Clearly this 'initial-segment-of' relation is a partial order. If $x \leq y$ and $x \neq y$ we write $x < y$.
(ii) Suppose $\mathcal{A}$ is totally ordered by a relation $<_{\mathcal{A}}$. Then we define $\preceq$ to be the *lexicographic ordering of* $\mathcal{A}^*$ *induced by* $<_{\mathcal{A}}$. That is, $x \preceq y$ if and only if $x \leq y$, or $x = zau$ and $y = zbv$ where $a, b \in \mathcal{A}$ and $a <_{\mathcal{A}} b$. Note that $\preceq$ totally orders $\mathcal{A}^*$.
(iii) For any $a \in \mathcal{A}$ we define $r_a : \mathcal{A}^* \to \mathcal{A}^*$, the *a-th successor function*, by $r_a(x) = xa$.

Definition 6.17 (**SnS**) For any n such that n is a natural number, or $n = \omega$, let T_n be $\{i \in \omega \mid i < n\}^*$. The structure $\mathfrak{N}_n$ is $(T_n, r_i, \leq, \preceq)_{i<n}$, where $\preceq$ is the lexicographic ordering induced by $<_\omega$, the usual ordering of the natural numbers. $\mathfrak{N}_n$ is called the *structure of* n *successor functions*. (Note that all these structures are countably infinite.)

The *monadic second-order theory of* n *successor functions* is the monadic second-order theory of $\mathfrak{N}_n$ in the monadic second-order language of appropriate signature (we spell out the details of this language below); this theory is usually referred to as SnS. $\dashv$

Let us spell out the intuitions underlying this machinery. First, note that each structure $\mathfrak{N}_n$ really is an infinite tree where each node has n immediate successors (or *daughters*, in standard tree terminology). For example, consider $\mathfrak{N}_1$; that is $(\{0\}^*, r_0, \leq, \preceq)$. This is the infinite tree in which each node has exactly one daughter; that is, it is simply an isomorphic copy of the natural numbers in their usual order. Next, consider $\mathfrak{N}_2$, that is $(\{0,1\}^*, r_0, r_1, \leq, \preceq)$. This is the full binary tree (that is, the infinite tree in which every node has exactly two daughters). An initial segment of $\mathfrak{N}_2$ is shown in Figure 6.1. Note that λ is the *root node* of the tree depicted in Figure 6.1, and that r_0 and r_1 are the *first daughter* and *second daughter* relations, respectively. Further, note that $\leq$ has a natural tree-geometric interpretation: it is simply the *dominates* relation. That is, $x \leq y$ iff it is possible to reach x by moving upwards in the tree from y. Similarly, $\preceq$ is the *dominates-or-to-the-left-of* relation. The tree-like nature of these models plays an important role in the work that follows, and must be properly understood. In particular, the reader should check that $\mathfrak{N}_\omega$ really is an infinite tree in which every node has ω daughters.

So much for the structures — what about the theories? Each of the theories SnS is a monadic *second-order* theory in the *appropriate* language. For example, the monadic second-order language appropriate for talking about $\mathfrak{N}_2$ contains two

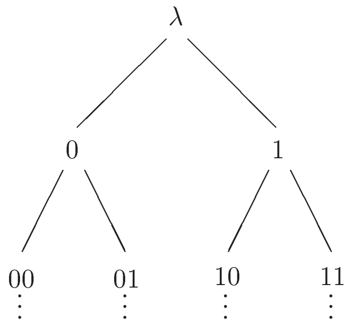


Fig. 6.1. An initial segment of $\mathfrak{N}_2$

function symbols for talking about r_0 and r_1 (we will be economical with our notation and use r_0 and r_1 for these symbols) and two binary predicate symbols for talking about $\leq$ and $\preceq$ (we use $\leq$ and $\preceq$ for this purpose). In addition, the language contains a denumerably infinite set of individual variables x, y, z, $\ldots$, a denumerably infinite set of predicate (or set) variables P, Q, S, $\ldots$, that range over subsets of the domain, and the usual quantifiers and boolean operators. The syntax and semantics of the language is standard; see Section A for further discussion of monadic second-order logic.

Using these languages, we can say many useful things about $\mathfrak{N}_n$. First, note that although we did not include a primitive equality predicate, an equality predicate is definable over $\mathfrak{N}_n$:

$$x = y \text{ iff } x \preceq y \wedge y \preceq x.$$

Next, note that we can define a unary predicate symbol ROOT that is true only of the root node λ:

$$\text{ROOT}(x) \text{ iff } \neg\exists y\,(y < x). \tag{6.1}$$

We can define the unary higher-order predicate 'P is a finite set.' Recall that a total ordering R on a set S is a *well-ordering* if every non-empty subset of S has an R-least element; it is a standard observation that S is well ordered by R iff S contains no infinitely descending R-chains. It follows that a subset P of T_n is finite iff it is well-ordered by both $\preceq$ and its converse, for such a set contains no infinitely descending $\preceq$-chains and no infinitely ascending $\preceq$-chains.

$$\begin{aligned} &\text{FINITE}(P) \text{ iff} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (6.2)\\ &\quad \forall Q\,((\exists x\, Qx \wedge \forall y\,(Qy \to Py)) \to \exists u\,(Qu \wedge \forall w\,(Qw \to u \preceq w))\\ &\qquad\quad \wedge\, \exists v\,(Qv \wedge \forall w\,(Qw \to w \preceq v))). \end{aligned}$$

(That is, P is finite if every non-empty subset of P has a $\preceq$-first and a $\preceq$-last element.) In short, monadic second-order logic is an extremely powerful language for talking about trees — which makes the following result all the more remarkable.

Theorem 6.18 (Rabin) *For any natural number n, or $n = \omega$,* SnS *is decidable.*

That is, for any n, it is possible to write a Turing machine which, when given a monadic second-order formula (in the language of appropriate signature), correctly decides whether or not the formula is satisfiable in $\mathfrak{N}_n$. The proof of this beautiful result is beyond the scope of this book; we refer the reader to the Notes for discussion and references.

Given a modal logic Λ, how can we use the fact of SnS-decidability to establish Λ-decidability? Suppose $\Lambda = \Lambda_{\mathsf{M}}$ for some class of *countable* models M. The essence of the interpretation method is to attempt to construct, for any modal formula ϕ, a monadic second-order formula *Sat*-$\Lambda(\phi)$ that does three things.

- It must encode the information in ϕ; this is usually achieved by using some variant of the standard translation.
- It must define a set of substructures of $\mathfrak{N}_n$ (for some choice of n) which are isomorphic copies of the models in M.
- It must bring the two previous steps together. That is, *Sat*-$\Lambda(\phi)$ must be constructed so that it is satisfiable in $\mathfrak{N}_n$ iff (the translation of) ϕ is satisfiable in (a definable substructure of $\mathfrak{N}_n$ that is isomorphic to) a model in M — that is, iff ϕ is Λ-satisfiable.

If such a formula *Sat*-$\Lambda(\phi)$ can be constructed, the ramifications for modal decidability are clear: as SnS is decidable, we can decide whether or not *Sat*-$\Lambda(\phi)$ is satisfiable on $\mathfrak{N}_n$. As this is equivalent to deciding the Λ-satisfiability of ϕ, we will have established that Λ is decidable.

As our first example of the method in action, we will prove the decidability of **KvB**. We met this logic briefly in Exercise 4.4.2; it is the logic of a certain general frame $\mathfrak{J}$. The domain J of $\mathfrak{J}$ consists of $\mathbb{N} \cup \{\omega, \omega + 1\}$ (that is, the set of natural numbers together with two further points), and the relation R is defined by Rxy iff $x \neq \omega + 1$ and $y < x$ or $x = \omega + 1$ and $y = \omega$. The frame (J, R) is shown in Figure 6.2. A, the collection of subsets of J admissible in $\mathfrak{J}$, consists of all $X \subseteq J$ such that either X is finite and $\omega \notin X$, or X is co-finite and $\omega \in X$.
As the reader was asked to show in Exercise 4.4.2, **KvB** is incomplete; that is, there is no class of *frames* F such that $\mathbf{KvB} = \Lambda_{\mathsf{F}}$. By Theorem 6.5 it follows that **KvB** lacks the finite model property. Even though it lacks the finite model property, **KvB** *is* decidable, and we will demonstrate this via an interpretation in S2S.

Theorem 6.19 **KvB** *is decidable.*

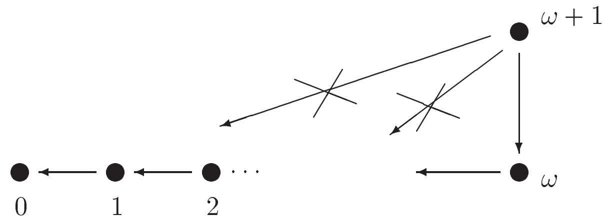


Fig. 6.2. The frame underlying $\mathfrak{J}$. Note that $\omega+1$ is related only to ω.

Proof. Let us make two initial assumptions; we will shortly show that both assumptions are correct. First, let us suppose that $\mathfrak{J} = (J, R, A)$ can be isomorphically embedded in $\mathfrak{N}_2$. We will refer to this isomorphic copy as $\mathfrak{J}$; no confusion should arise because of this double usage. Furthermore, let us suppose that this isomorphic image is *definable* in the monadic second-order language for $\mathfrak{N}_2$. That is, suppose that there are formulas $\widehat{J}(x)$, $\widehat{R}(x,y)$ and $\widehat{A}(P)$ (containing 1 free individual variable x, 2 free individual variables x and y, and 1 free predicate variable P, respectively) such that

$$\begin{array}{rcl} J & = & \{t \in T_2 \mid \mathfrak{N}_2 \models \widehat{J}(x)[t]\} \\ R & = & \{(t,t') \in T_2 \times T_2 \mid \mathfrak{N}_2 \models \widehat{R}(x,y)[t,t']\} \\ A & = & \{U \subseteq T_2 \mid \mathfrak{N}_2 \models \widehat{A}(P)[U]\}. \end{array}$$

Given these assumptions, it is easy to reduce the satisfiability problem for the general frame to the satisfiability problem for S2S. First, with the help of the formula $\widehat{R}$, we can define a translation T from the modal language into the second-order language:

$$\begin{array}{rcl} T_x(p) & = & Px \\ T_x(\neg\phi) & = & \neg T_x(\phi) \\ T_x(\phi \wedge \psi) & = & T_x(\phi) \wedge T_x(\psi) \\ T_x(\Diamond\phi) & = & \exists y\,(\widehat{R}xy \wedge T_y(\phi)) \end{array}$$

Note that T_x is just the standard translation with the formula $\widehat{R}$ replacing the use of a fixed relation symbol. We leave it as an exercise to show that for any modal formula ϕ (built out of proposition letters $p_1, \ldots, p_n$)

$$((J,R,A),V), w \Vdash \phi \text{ iff } \mathfrak{N}_2 \models T_x(\phi)[w, V(p_1), \ldots, V(p_n)]. \tag{6.3}$$

(See Exercise 6.3.1; the notation $[w, V(p_1), \ldots, V(p_n)]$ means assign the state w to the free variable x, and assign the subset $V(p_i)$ to the predicate variable P_i.)

For any modal formula ϕ, let *Sat-KvB*(ϕ) be the following monadic second-order sentence:

$$\exists P_1 \ldots \exists P_n \exists x\,(\widehat{A}(P_1) \wedge \cdots \wedge \widehat{A}(P_n) \wedge \widehat{J}(x) \wedge T_x(\phi)).$$

It follows that ϕ is satisfiable in (J, R, A) iff $\mathfrak{N}_2 \models$ *Sat-KvB*(ϕ). Thus — *given our two initial assumptions* — we have effectively reduced the satisfiability problem for **KvB** to the S2S-satisfiability problem, for ϕ is satisfiable on $\mathfrak{J}$ iff *Sat-KvB*(ϕ) belongs to S2S, and by Rabin's result it is possible to decide the latter.

Hence, to complete the proof that **KvB** is decidable, it only remains to show that our assumptions were justified; that is, to show that $\mathfrak{J}$ really does have a definable isomorphic image in $\mathfrak{N}_2$. Given the expressive power at our disposal, this is actually rather easy to do. We will make use of the general predicates $=$, ROOT, and FINITE defined in (6.1) and (6.2). In addition, we will use

$$x <_1 y \text{ iff } r_1(x) \leq y \wedge \neg \exists z (x \leq z \wedge r_0(z) \leq y).$$

Note that $x <_1 y$ means that x is a proper initial subsequence of y such that y extends x by a finite sequence of 1s — or, in terms of tree geometry, it is possible to move down from x to y by using only the 'second daughter' relation.

We will now define an isomorphic image of $\mathfrak{J}$ in $\mathfrak{N}_2$. First, we can define the numeric part of the underlying frame as follows:

$$N(x) \text{ iff } \textsc{Root}(x) \vee \exists y\, (\textsc{Root}(y) \wedge y <_1 x).$$

The isomorphism involved should be clear: the natural number zero is taken to be the empty sequence, and the positive integer n is taken to be the sequence of n 1s. Next, we will represent ω by 0, and $\omega + 1$ by 00. Defining these choices is easy:

$$\begin{array}{rcl} \textsc{Omega}(x) & \text{iff} & \exists y\, (\textsc{Root}(y) \wedge x = r_0(y)) \\ \textsc{Omega+1}(x) & \text{iff} & \exists y\, (\textsc{Root}(y) \wedge x = r_0(r_0(y))). \end{array}$$

Putting it all together, we define the required predicates $\widehat{J}$ and $\widehat{R}$ as follows:

$$\begin{array}{rcl} \widehat{J}(x) & = & Nx \vee \textsc{Omega}(x) \vee \textsc{Omega+1}(x) \\ \widehat{R}(x,y) & = & (Ny \wedge (y <_1 x \vee \textsc{Omega}(x))) \vee (\textsc{Omega}(y) \wedge \textsc{Omega+1}(x)). \end{array}$$

Clearly these two formulas define a subset of the tree domain isomorphic to (J, R). Thus it merely remains to define A, the class of allowable valuations. With the help of the FINITE predicate, this is straightforward.

$$\begin{array}{l} \widehat{A}(P) \text{ iff} \\ \quad \forall x\, (Px \to \widehat{J}(x)) \wedge ((\textsc{Finite}(P) \wedge \forall z\, (\textsc{Omega}(z) \to \neg Pz)) \vee \\ \qquad \forall Q \forall x\, (Qx \leftrightarrow (Jx \wedge \neg Px) \to (\textsc{Finite}(Q) \wedge \forall z\, (\textsc{Omega}(z) \to \neg Qz)))) \end{array}$$

In short, a definable isomorphic image of $\mathfrak{J}$ really does live inside $\mathfrak{N}_2$. We conclude that **KvB** is decidable. $\dashv$

While the above result is a nice introduction to decidability via interpretation, in one respect it is rather misleading. **KvB** is characterized by a single structure (and a rather simple one at that) thus we only had to define a single isomorphic image, and were able to do this fairly straightforwardly using S2S. However, as we saw in Chapter 4, it is usual to characterize logics in terms of a *class* of structures; for example, **S4** is usually characterized as the logic of the class of reflexive and transitive models. Do class-based characterizations mesh well with the idea of decidability via interpretations? Classes of models may contain uncountable structures — and only countable structures can be isomorphically embedded in $\mathfrak{N}_n$. And why should we expect to be able to isomorphically embed even countable models in infinite *trees*?

Two simple observations clear the way. First, in many important cases, only the *countable* structures in characterizing classes are required. Second, there is a standard method for converting a model into a tree-based model, namely the *unraveling* method studied in Chapters 2 and 4. Taken together, these observations enable us to view the classes of structures characterizing many important logics as a collection of definable substructures of $\mathfrak{N}_\omega$. We will illustrate the key ideas involved by proving the decidability of **S4** via interpretation in SωS.

As a first step, we claim that **S4** is sound and strongly complete with respect to the class of *countable* reflexive and transitive models. We could prove this directly (for example, using the step-by-step method discussed in Section 4.6) but it also follows from the following general observation. (Recall that for the duration of this chapter, we are only working with countable languages.)

Theorem 6.20 *If Λ is a normal logic that is sound and strongly complete with respect to a first-order definable class of models* M*, then Λ is also sound and strongly complete with respect to the class of all* countable *models in* M.

Proof. Left as Exercise 6.3.3. $\dashv$

Lemma 6.21 **S4** *is sound and strongly complete with respect to the class of countable (reflexive and transitive) trees.*

Proof. By Theorems 4.29 and 6.20, **S4** is sound and strongly complete with respect to the class of *countable* reflexive, transitive models; that is, every **S4**-consistent set of sentences Σ is satisfiable on such a model $\mathfrak{M} = (W, R, V)$ at some point w. Now, (as in the proof of Theorem 4.54) let $\vec{\mathfrak{M}} = (\vec{W}, \vec{R}, \vec{V})$ be the unraveling of $\mathfrak{M}$ around w, and let $\mathfrak{M}^*$ be $(\vec{W}, R^*, \vec{V})$, where R^* is the reflexive transitive closure of $\vec{R}$; this model is a reflexive transitive tree that verifies Σ at its root. Moreover, it is a *countable* model, for its nodes are all the finite sequences of states in $\mathfrak{M}$ that start at w, and as $\mathfrak{M}$ is countable, there are only countably many such sequences. The result follows. $\dashv$

Corollary 6.22 **S4** *is decidable, and its decidability can be proved via interpretations.*

Proof. Let us call a subset of $\mathfrak{N}_\omega$ an *initial subtree* if it contains λ and is closed under the inverse of $\leq$ (that is, if y belongs to the subset, and $x \leq y$, then x belongs to the subset). If S is such a subtree, then $\leq_S$ denotes the restriction of $\leq$ to S. Now for the key observation. Let $(\vec{W}, \vec{R})$ be the unraveling of some *countable* **S4**-frame (W, R) around a point w, and let R^* be the reflexive transitive closure of $\vec{R}$. Then $(\vec{W}, R^*)$ is isomorphic to a pair $(S, \leq_S)$ for some initial subtree S. To see this, note that we can inductively construct an isomorphism f from $(\vec{W}, R^*)$ to some initial subtree as follows. First, we stipulate that f maps the root of $(\vec{W}, R^*)$ to λ. Next, suppose that for some $\vec{u} \in \vec{W}$, $f(\vec{u})$ has been defined to be m. Now, $\{\vec{s} \in \vec{W} \mid \vec{u}\vec{R}\vec{s}\}$ is a countable set as $\vec{W}$ is countable, so we can enumerate its elements. Then, if $\vec{s}$ is the i-th element in this enumeration, we stipulate that $f(\vec{s}) = r_i(m)$. (That is, the successor of $\vec{u}$ that is i-th in our enumeration is mapped to the i-th successor of m.) In short, $\mathfrak{N}_\omega$ is 'wide enough' to accommodate a copy of every branch through a tree-like **S4** model in a very obvious way. In fact, it is precisely because the required isomorphisms are so simple that we have elected to work with $\mathfrak{N}_\omega$.

With this observed, the interpretation is easy to define. First, we define a predicate $\textsc{Isubtree}(S)$, which picks out the initial subtrees of $\mathfrak{N}_\omega$:

$$\textsc{Isubtree}(S) \text{ iff } \exists y\,(\textsc{Root}(y) \wedge Sy) \wedge \forall z \forall u\,((Sz \wedge u \leq z) \rightarrow Su)).$$

Second, we define a predicate $\leq_S$ that defines the restriction of $\leq$ to a subset S of $\mathfrak{N}_\omega$ by

$$x \leq_S y \text{ iff } Sx \wedge Sy \wedge x \leq y.$$

Third, we define a translation T^ω from the basic modal language to the monadic second-order language for $\mathfrak{N}_\omega$. Like the translation T we used when proving the decidability of **KvB** this translation is a simple variant of the standard translation. In fact, it is identical to T save in the clause for modalities, which is given by:

$$T^\omega_{x,S}(\Diamond\phi) = \exists y\,(x \leq_S y \wedge T^\omega_{y,S}(\phi)).$$

Note that as well as containing the free individual variable x, the translation of $\Diamond\phi$ contains a free set variable S; when written in full the above expression becomes:

$$T^\omega_{x,S}(\Diamond\phi) = \exists y\,(Sx \wedge Sy \wedge x \leq y \wedge T^\omega_{y,S}(\phi)).$$

We need the free variable here because we are not working with one fixed isomorphic image (as we were when proving the decidability of **KvB**). Rather, we have a separate relation for each initial subtree, and the presence of the free variable allows all our definitions to be relativized in the appropriate way.

It simply remains to put it all together. Suppose ϕ is a modal formula constructed out of the proposition letters $p_1, \ldots, p_n$. Define *Sat-S4*(ϕ) to be the following sentence:

$$\exists S \exists P_1 \ldots \exists P_n \exists x \big(\textsc{Isubtree}(S) \wedge \\ \forall z\,(P_1 z \to Sz) \wedge \cdots \wedge \forall z\,(P_n z \to Sz) \wedge Sx \wedge T^{\omega}_{x,S}(\phi)\big).$$

Recall that $T^{\omega}(\phi)$ contains free occurrences of S and x; these become bound in this sentence. Bearing this in mind, it is clear this sentence asserts the existence of an initial subtree S of $\mathfrak{N}_\omega$, a collection of n subsets P_i of this subtree, and a state x in the subtree, that satisfy the translation of ϕ. That is, it asserts the existence of a tree-like **S4** model for the (translation of) ϕ, and we have reduced the **S4**-satisfiability problem to the SωS-satisfiability problem. $\dashv$

This completes our discussion of interpretations in SnS — though we should immediately admit that we have barely scratched the surface of the method's potential: Rabin's theorem is very strong, the ideas underlying it make contact with many branches of mathematics, and it has become a fundamental tool in many branches of logic and theoretical computer science. Nonetheless, our discussion has unearthed themes relevant to modal logic: the importance of establishing completeness results with respect to classes of *countable* structures, the use of *unraveling* to produce tree-like models, and the particular utility of $\mathfrak{N}_\omega$ in allowing reasonably straightforward isomorphic embeddings. These three ideas enable a wide range of modal decidability results to be proved via interpretations.

One final remark: while SnS is important, it is certainly not the only logical system in which modal logics can be interpreted. Many fragments of classical logic, or theories in classical logics, are known to be decidable, and offer opportunities for proving modal decidability results. Indeed we have already met a (very simple) example. We pointed out in Section 2.4 that the basic modal language translates into the 2 variable fragment of classical logic, (see Proposition 2.49), from which it immediately follows that **K** (and some simple extensions such as **T**) are decidable. Moreover, on occasions it can be useful to interpret a modal logic in another modal logic already known to be decidable. See the Notes for further discussion.

Exercises for Section 6.3

6.3.1 We claimed that the general frame for **KvB** is isomorphically embedded in the tree domain, and that $\widehat{R}$ defines the accessibility relation of this isomorphic image. Check this claim, and show that

$$((W,R,A),V),w \Vdash \phi \text{ iff } \mathfrak{N}_2 \models T_x(\phi)[w, V(p_1),\ldots,V(p_n)],$$

for any modal formula ϕ (see (6.3)).

6.3.2 Show by interpretation in S2S that both the tense logic of the natural numbers, and the tense logic of the integers, are decidable. Now add the until operator U to your language (this operator was defined in Chapter 2 in Exercise 2.2.4). Are the logics of the natural numbers and the integers in this richer language still decidable?

6.3.3 Prove Theorem 6.20. That is, show that if Λ is a normal logic that is sound and strongly complete with respect to a first-order definable class of models M, then Λ is also sound and strongly complete with respect to the class of all *countable* models in M. (Hint: use the standard translation and the Downward Löwenheim-Skolem Theorem.)

6.4 Decidability via Quasi-models and Mosaics

In this section we will show that such familiar techniques as filtration can be employed to prove decidability, even for logics lacking the finite model property. The key move is simply to think more abstractly: instead of trying to work with finite models themselves, we will work with finite structures which encode information *about* models.

Quasi-models for $\mathbf{KvB}$

For our first example we will re-examine the logic $\mathbf{KvB}$, which we proved decidable in the previous section via interpretation in S2S. Recall that $\mathbf{KvB}$ is the logic of a single general frame $\mathfrak{J}$ whose universe J is $\mathbb{N} \cup \{\omega, \omega + 1\}$, and whose accessibility relation is R. Also recall that $\mathbf{KvB}$ is an *incomplete* logic, which implies that it does *not* have the finite model property. Nonetheless, we can establish the decidability of $\mathbf{KvB}$ using a filtration argument. We cannot use filtration to build a finite $\mathbf{KvB}$ *model* (no such model exists), but we can use it to build a finite *quasi-model*.

Consider a model $\mathfrak{M} = (J, R, V)$, where V is an admissible valuation for $\mathfrak{J}$. What kind of filtration seems natural for this structure? If it were not for the point $\omega + 1$, it is obvious that we would go for the transitive filtration. Very well then — let's adopt the following procedure: first delete the point $\omega + 1$, then take the transitive filtration of the remainder of the frame, and finally glue a copy of the point $\omega + 1$ back on to the resulting finite structure. Of course, we know that this will not result in a finite $\mathbf{KvB}$ model; but hopefully it will yield something from which we can *construct* a $\mathbf{KvB}$ model.

First we need the notion of a *closure* of a set of sentences. We will not filtrate through arbitrary subformula-closed sets of sentences; rather, we will insist on working with sets of sentences that are closed under single negations as well.

Definition 6.23 (Closed Sets and Closures) A set of formulas Σ is said to be *closed* if it is closed under subformulas and single negations. That is, if $\sigma \in \Sigma$ and θ is a subformula of σ, then $\theta \in \Sigma$; and moreover if $\sigma \in \Sigma$, and σ is not of the form $\neg\theta$, then $\neg\sigma \in \Sigma$.

If Γ is a set of formulas, then $Cl(\Gamma)$, the *closure* of Γ, is the smallest closed set of formulas containing Γ. Note that if Γ is finite then so is $Cl(\Gamma)$. If $\Gamma = \{\phi\}$, where ϕ is any modal formula, then we usually write Cl_ϕ for $Cl(\{\phi\})$ and call this set the closure of ϕ. $\dashv$

Advanced track readers should note that they have already met a more elaborate version of this idea when we proved the completeness of **PDL**: any Fisher-Ladner closed set (see Definition 4.79) is closed in the sense of the previous definition.

Now for quasi-models. Let ϕ be some basic modal formula. A $\mathbf{KvB}$ *quasi-model for* ϕ is a pair $\mathfrak{Q} = (\mathfrak{F}, \lambda)$ where:

(i) $\mathfrak{F} = (Q, S)$ is a finite frame, containing two distinct distinguished points called c and ∞, that satisfies conditions F1–F5 below; and
(ii) λ is a function mapping states of $\mathfrak{F}$ to subsets of Cl_ϕ that satisfies the conditions L0–L3 below. We call λ a *labeling*.

Let's first consider the conditions F1–F5. These are very simple, and should be checked against Figure 6.2. If you read c as 'co-finite' and view this element as the quasi-model's analog of ω, and view ∞ as the analog of $\omega + 1$, the resemblance between finite frames fulfilling these conditions and the frame (J, R) should be clear.

(F1) On $Q \setminus \{\infty\}$, S is trichotomous and transitive,
(F2) Scw iff $w \neq \infty$,
(F3) Swc iff $w = c$ or $w = \infty$,
(F4) $S\infty w$ iff $w = c$, and
(F5) $Sw\infty$ for no w in Q.

Note that c is *reflexive*. Intuitively, the filtration process described above squashes ω down into a cluster.

There are also conditions on the labeling. One of these conditions is that every label should be a *Hintikka set*. This is an important concept, and one we will use again later in this chapter.

Definition 6.24 (Hintikka Sets) Let Σ be a closed set of formulas. A *Hintikka set* H over Σ is a maximal subset of Σ that satisfies the following conditions:

(i) $\bot \notin H$
(ii) If $\neg\phi \in \Sigma$, then $\neg\phi \in H$ iff $\phi \notin H$.
(iii) If $\phi \wedge \psi \in \Sigma$, then $\phi \wedge \psi \in H$ iff $\phi \in H$ and $\psi \in H$. $\dashv$

It is important to realize that Hintikka sets also satisfy conditions such as the following: if $\phi \vee \psi \in \Sigma$, then $\phi \vee \psi \in H$ iff $\phi \in H$ or $\psi \in H$. This is because in this book we define $\vee$ (and also $\rightarrow$, $\leftrightarrow$, and $\top$) in terms of $\bot$, $\neg$, and $\wedge$ (see Definition 1.12). Hintikka sets need *not* be satisfiable (the reader is asked to construct a non-satisfiable Hintikka set in Exercise 6.4.2) but items (i) and (ii) above guarantee that they contain no blatant propositional contradictions. If a Hintikka set is satisfiable we call it an *atom*. Note that both the MCSs used to build canonical models, and the special atoms used to prove the completeness of **PDL** are examples of (consistent) Hintikka sets.

We are now ready for the quasi-model labeling conditions:

(L0) $\phi \in \lambda(w)$ for some $w \in Q$,
(L1) $\lambda(w)$ is a Hintikka set, for each $w \in Q$,
(L2) For all $\Diamond\psi \in Cl_\phi$, $\Diamond\psi \in \lambda(w)$ iff $\psi \in \lambda(v)$ for some v with Swv,
(L3) If $\Diamond\psi \in \lambda(w)$, then $\psi \in \lambda(v)$ for some v with Swv *and not* Svw.

We take the *size* of a quasi-model (Q, S, λ) to be the size of its universe Q.

Lemma 6.25 *Let ϕ be a formula in the basic modal language. Then ϕ is satisfiable in $\mathfrak{J}$ if and only if there is a quasi-model for ϕ, of size at most* $2^{|\phi|}$.

Proof. We leave it to the reader to prove the left to right direction; this is simply a matter filling in the details of the 'delete $\omega + 1$, filtrate, glue $\omega + 1$ back on' strategy sketched above (the filtration must be made through Cl_ϕ) and the upper bound on the size of the quasi-model follows as in any filtration argument. So let's look at the right to left direction.

Let $\mathfrak{Q} = (Q, S, \lambda)$ be a quasi-model for ϕ, let c and ∞ be the distinguished points of the quasi-model, and let Q_0 denote the set $Q \setminus \{\infty\}$. We now define an equivalence relation $\sim_0$ on Q_0 by

$$w \sim_0 v \text{ iff } w = v \text{ or } (Swv \text{ and } Svw).$$

This really is an equivalence relation, and a more-or-less familiar one at that: the equivalence class $\bar{w}$ containing a *reflexive* point w is simply the *cluster* that w belongs to (see Definition 4.55), while the equivalence class $\bar{w}$ containing an *irreflexive* point w is simply $\{w\}$. The equivalence classes on Q_0 are naturally ordered by the relation $\succ$ defined as follows:

$$\bar{w} \succ \bar{v} \text{ iff } Swv \text{ and not } Svw.$$

It follows from F1 that $\succ$ is a strict total ordering. Now consider an enumeration $q_0, q_1, \ldots, q_N$ of the elements of Q_0, such that q first enumerates all elements of the leftmost equivalence class, then all elements of its rightmost neighbor, and so on. We may extend this enumeration to a map $f : J \to Q$ by putting

$$f(w) = \begin{cases} q_w & \text{if } w \leq N, \\ c & \text{if } w > N \text{ or } w = \omega, \\ \infty & \text{if } w = \omega + 1. \end{cases}$$

It is straightforward to check that for all w, v in J, Rwv implies $Sf(w)f(v)$. Consider, for instance, the case where $w = \omega$; Rwv implies that $v = n$ for some natural number n. But then $f(w) = c$ and $f(v) \in Q_0$, so $Sf(w)f(v)$ follows from F2. The other cases are left to the reader. What we have shown is that

$$f \text{ is a homomorphism mapping } (J, R) \text{ onto } (Q, S). \tag{6.4}$$

Now consider the following valuation V on (J, R):

$$V(p) = \{w \in J \mid p \in \lambda(f(w))\}.$$

It is easy to see that V is admissible in the general frame $\mathfrak{J}$: if $\omega \in V(p)$ then by definition of V, $n \in V(p)$ for all $n > N$, so $V(p)$ is co-finite.

Hence, in order to prove the lemma, it is sufficient to show that ϕ holds somewhere in the model $(\mathfrak{J}, V)$; but this follows from L0 and the following claim:

$$\text{for all } \psi \in Cl_\phi, \text{ and all } w \in J: \quad \mathfrak{J}, V, w \Vdash \psi \text{ iff } \psi \in \lambda(f(w)). \tag{6.5}$$

We will prove this claim by induction on the complexity of ψ. The base case, where ψ is a propositional variable, holds by definition of V, and the induction step for the boolean connectives is trivial since λ labels with *Hintikka sets* only. Hence, the only interesting case is where ψ is of the form $\Diamond\chi$. Note that the inductive hypothesis applies to χ and that $\chi \in Cl_\phi$ since the set is closed under taking subformulas.

First assume that $\mathfrak{J}, V, w \Vdash \Diamond\chi$. There is a state v with Rwv and $v \Vdash \chi$. By the fact that f is a homomorphism it is immediate that $Sf(w)f(v)$, while the inductive hypothesis implies that $\chi \in \lambda(f(v))$. From this and L2 it follows that $\Diamond\chi \in \lambda(f(w))$.

Now suppose, in order to prove the other direction of (6.5), that $\Diamond\chi \in \lambda(f(w))$. We have to show that $\mathfrak{J}, V, w \Vdash \Diamond\chi$. Distinguish the following cases:

(i) $w = \omega + 1$. From this it follows that $f(w) = \infty$, so from L2 and the fact (F4) that c is the *only* successor of ∞, it follows that $\chi \in \lambda(c)$. Hence from $c = f(\omega)$ and the inductive hypothesis it follows that $\omega \Vdash \chi$. But then it is immediate that $\omega + 1 \Vdash \Diamond\chi$.

(ii) $w \neq \omega + 1$. By L3 we may assume the existence of an element $q \in Q$ satisfying $Sf(w)q$, not $Sqf(w)$ and $\chi \in \lambda(q)$. It is obvious from $Sf(w)q$ and $F5$ that $q \neq \infty$. Let v be a pre-image of q; from $q \neq \infty$ it follows that $v \neq \omega + 1$. Since R is trichotomous on $J \setminus \{\omega + 1\}$, we have Rvw or $w = v$ or Rwv. The first two options are impossible: Rvw would imply $Sf(v)f(w)$, while $w = v$ is incompatible with the fact that $Sf(v)f(w)$ but not $Sf(w)f(v)$. Hence, we find that Rwv; but the induction hypothesis gives $v \Vdash \chi$, so indeed we have $w \Vdash \Diamond\chi$.

This finishes the proof of (6.5) and hence, of the lemma. $\dashv$

Theorem 6.26 *The logic* $\mathbf{KvB}$ *is decidable.*

Proof. By Lemma 6.25 it suffices to show that it is decidable whether there is a quasi-model for ϕ of size not exceeding $2^{|\phi|}$. But this is easy to see: we first make a finite list of all triples (Q, S, λ) such that $|Q| \leq 2^{|\phi|}$, $S \subseteq Q \times Q$ and $\lambda : Q \to \mathcal{P}(Cl_\phi)$; we then check for each member of this list whether it is a quasi-model for ϕ. And clearly it is possible to write a terminating program which does this. $\dashv$

The important lesson is that in order to prove decidability of a logic, not only finite *models* are useful: rather, any finite structure that *encodes* a model is potentially valuable. Now, the finite structure employed in the previous example was still very much *like* a model — as our name 'quasi-model' indicates — but in the general case one can push the idea much further. The satisfiability of ϕ doesn't need to be witnessed by a finite model for ϕ, or indeed by anything that looks very much like a model; all we need is a finite *toolkit* which contains the instructions needed to construct a model for ϕ. The concept of a *mosaic* develops this line of thought.

Mosaics for the tense logic of the naturals

Consider the frame $\mathfrak{N} = (\mathbb{N}, <)$ with $<$ the standard ordering of the natural numbers, and let $\mathbf{K}_t\mathbf{N}$ be its tense logic. $\mathbf{K}_t\mathbf{N}$ does *not* have the finite model property (see Exercise 6.4.1), but it *is* decidable, as we will now show using mosaics.

We use the following terminology and notation. For a given formula ϕ in the basic temporal similarity type, let Cl_ϕ denote the smallest subformula closed set containing ϕ (note that we use the same notation as before, but for a different set since we are dealing with a different similarity type).

Definition 6.27 (Bricks) A *brick* is a pair $b = (\Phi, \Lambda)$ such that Φ and Λ are Hintikka sets satisfying

(B0) if $G\psi \in \Phi$, then $G\psi, \psi \in \Lambda$,
(B1) if $H\psi \in \Lambda$, then $H\psi, \psi \in \Phi$.

A brick is called *small* if it satisfies, in addition:

(B2) if $F\psi \in \Phi$, then either ψ or $F\psi$ is in Λ,
(B3) if $P\psi \in \Lambda$, then either ψ or $P\psi$ is in Φ.

What we are really interested in are *sets* of bricks satisfying certain saturation conditions. A brick set B is a *saturated set of bricks for* ϕ (in short: a ϕ-SSB) if it satisfies

(S0) for some (Φ, Λ), $H\bot \in \Phi$ and $\phi \in \Lambda \cup \Phi$,
(S1) for all $(\Phi, \Lambda) \in B$, if $F\psi \in \Lambda$ then there is a $(\Lambda, \Gamma) \in B$ with $\psi \in \Gamma$,
(S2) for all $(\Phi, \Lambda) \in B$ there is a path of small bricks leading from Φ to Λ.

Here we say that a *path* of (small) bricks from Φ to Λ is a sequence $(\Phi_0, \Lambda_0), \ldots, (\Phi_n, \Lambda_n)$ $(n \geq 0)$ of (small) bricks such that $\Phi = \Phi_0$, $\Lambda = \Lambda_n$ and $\Lambda_{i+1} = \Phi_i$ for all $i < n$. Finally, we simply define the *size* of an SSB B to be the number of bricks in B. $\dashv$

The best way of grasping the intuitive meaning of these notions is by reading the proof of the next lemma.

Lemma 6.28 *If ϕ is satisfiable in $\mathfrak{N}$, then there is a ϕ-SSB of size at most $2^{2|\phi|}$.*

Proof. Assume that we have a valuation V on $\mathfrak{N}$ such that ϕ is true at the number k. For any number n, let Γ_n denote the truth set of n:

$$\Gamma_n = \{\psi \in Cl_\phi \mid \mathfrak{N}, V, n \Vdash \psi\}.$$

Define B as the set

$$B = \{(\Gamma_n, \Gamma_m) \mid n < m\},$$

and call a brick *sequential* if it is of the form (Γ_n, Γ_{n+1}) for some number n. We now prove that B satisfies the conditions B0–B3 and S0–S2.

For B0, assume that $G\psi \in \Gamma_n$ and that $n < m$; we have to prove that both $G\psi$ and ψ belong to Γ_m. But from the assumption it follows that $\mathfrak{N}, V, n \Vdash G\psi$. This implies that $n' \Vdash \psi$ for all $n' > n$; in particular, $m \Vdash \psi$. But also $m \Vdash G\psi$, by transitivity of $<$. By definition of Γ_m then we have $G\psi \in \Gamma_m$ and $\psi \in \Gamma_m$. B1 is proved in a similar way.

For B2, take an arbitrary sequential brick (Γ_n, Γ_{n+1}) and assume that $F\psi \in \Gamma_n$. By definition, $\mathfrak{N}, V, n \Vdash F\psi$, so there must be some $m > n$ with $m \Vdash \psi$. Note that either $m = n + 1$ or $m > n + 1$; in the first case, we obtain $\psi \in \Gamma_{n+1}$, in the second, $F\psi \in \Gamma_{n+1}$. B3 is proved similarly, hence all sequential bricks are small.

It is likewise straightforward to prove the saturation conditions. For example, in order to prove S0, we consider the brick (Γ_0, Γ_k) (recall that k is the state where ϕ holds).

Finally, the collection $\{\Gamma_n \mid n \in \mathbb{N}\}$ is a subset of the power set of Cl_ϕ, whence its cardinality does not exceed $2^{|\phi|}$; but then the size of B can be at most $(2^{|\phi|})^2 = 2^{2|\phi|}$. $\dashv$

We now show that we have recorded enough information in the definition of a saturated set of bricks for ϕ to construct an $\mathfrak{N}$-based model for ϕ.

Lemma 6.29 *If there is a ϕ-SSB, then ϕ is satisfiable in $\mathfrak{N}$.*

Proof. Assume that B is a saturated set of bricks for ϕ. We will use these bricks to build, step by step, the required model for ϕ. As usual, in each finite stage of the construction we are dealing with a finite approximation of this model: a *history* is a pair (L, λ) such that L is a natural number and λ is a function on the set $\{0, \ldots, L\}$ to the set of atoms. Such a history (L, λ) is supposed to satisfy the following constraints:

(H0) $H\bot \in \lambda(0)$.
(H1) for all m with $m < L$, $(\lambda(m), \lambda(m+1))$ is a small brick.

We leave it to the reader to verify that any history (L, λ) has the following properties:

(H2) if $F\psi \in \lambda(n)$ for some $n < L$, then there is some m with $n < m \leq L$ and $\psi \in \lambda(m)$, or otherwise $F\psi \in \lambda(L)$.
(H3) if $P\psi \in \lambda(n)$ for some $n \leq L$, then there is some m with $m < n$ and $\psi \in \lambda(m)$.

The importance of the properties H2 and H3 is that they show that the only essential shortcomings of a history (regarded as a finite approximation of a model) are of the form '$F\psi \in \lambda(L)$, and there is no witness for this fact; that is, no $m > L$ such that $\psi \in \lambda(m)$.'

Of course, we are not going to use histories in isolation; we say that one history (L', λ') is an *extension* of another history (L, λ), notation: $(L, \lambda) \lhd (L', \lambda')$, if $L < L'$, while λ and λ' agree on the domain of λ. The crucial extension lemma of the step-by-step construction is given in the following claim.

$$\begin{array}{l}\text{Any history } (L, \lambda) \text{ with } F\psi \in \lambda(L) \\ \text{has an extension } (L', \lambda') \text{ with } \psi \in \lambda'(L').\end{array} \tag{6.6}$$

To prove (6.6), let (L, λ) be a history and ψ a formula such that $F\psi \in \lambda(L)$. It follows from H1 that $(\lambda(L-1), \lambda(L))$ is a brick, so by S1 there is a brick (Φ, Λ) in B such that $\Phi = \lambda(L)$ and $\psi \in \Lambda$. We now use S2 to find a path of small bricks (Σ_0, Σ_1), (Σ_1, Σ_2), ..., (Σ_{k-1}, Σ_k), such that $\Sigma_0 = \Phi$ and $\Sigma_k = \Lambda$. Obviously we are going to 'glue' this path to the old history, thus creating a new history (L', λ'). To be precise, L' is defined as $L' = L + k$, while λ' is given by

$$\lambda'(n) = \begin{cases} \lambda(n) & \text{if } n \leq L \\ \Sigma_i & \text{if } n = L + i. \end{cases}$$

With this definition (L', λ') satisfies the condition of (6.6).

Using (6.6), by a standard step by step construction one can define a sequence $(L_0, \lambda_0) \lhd (L_1, \lambda_1) \lhd \ldots$ of histories such that $H\bot \in \lambda_0(0)$ and $\phi \in \lambda_0(L_0)$, while for each i and each formula $F\psi \in \lambda_i(L_i)$ there is a $j > i$ and a number $L_i < m \leq L_j$ such that $\psi \in \lambda_j(m)$. This sequence of nested histories will be our guideline for the definition of a valuation on $\mathfrak{N}$. Note that for all formulas ψ and all i and n, we have that

$$\text{if } n \leq L_i, \text{ then } \psi \in \lambda_i(n) \text{ iff } \psi \in \lambda_j(n) \text{ for all } j \geq i.$$

In other words, the histories always agree where they are defined; this fact will be used below without explicit comment.

Now consider the following valuation V on $\mathfrak{N}$:

$$V(p) = \{n \in \mathbb{N} \mid p \in \lambda_i(n) \text{ for some } i\}.$$

We are now ready to prove the crucial claim of this lemma.

$$\text{For all } \psi \in Cl_\phi \text{ and all } n\text{: } \mathfrak{N}, V, n \Vdash \psi \text{ iff } \psi \in \lambda_i(n) \text{ for some } i. \qquad (6.7)$$

Obviously, (6.7) will be proved by induction on ψ. The base step and the boolean cases of the induction step are straightforward and we leave them to the reader; we concentrate on the modal cases.

First assume that ψ is of the form $F\chi$. For the direction from left to right, assume that $\mathfrak{N}, V, n \Vdash F\chi$. There must be a number $m > n$ with $m \Vdash \chi$; so by the inductive hypothesis, there is an i with $\chi \in \lambda_i(m)$. It is easy to show (by backward induction and H1) that this implies $F\chi \in \lambda_i(k)$ for all k with $n \leq k < m$.

For the other direction, assume that $F\chi \in \lambda_i(n)$ for some i. It follows from H2 that there is either a number m with $n < m \leq L_i$ and $\chi \in \lambda_i(m)$, or otherwise $F\chi \in \lambda_i(L_i)$. In the first case we use the inductive hypothesis to establish that $m \Vdash \chi$ and hence, $n \Vdash F\chi$. Hence, assume that we are in the other case: $F\chi \in \lambda_i(L_i)$. Now our sequence of histories is such that this implies the existence of a history (L_j, λ_j) with $j > i$ and such that $\chi \in \lambda_j(m)$ for some m with $L_i, m \leq L_j$. It follows from the inductive hypothesis that $m \Vdash \chi$; thus the truth definition gives us that $n \Vdash F\chi$.

Now assume that ψ is of the form $P\chi$. The direction from left to right is as in the previous case. For the other direction, assume that $P\chi \in \lambda_i(n)$ for some i; it follows by H3 that there is an $m < n$ with $\chi \in \lambda_i(m)$. The inductive hypothesis yields that $\mathfrak{M}, V, m \Vdash \chi$, so by the truth definition we get $n \Vdash F\chi$. $\dashv$

Theorem 6.30 $\mathbf{K_tN}$ *is decidable*

Proof. Immediate by Lemmas 6.28 and 6.29, and the obvious fact that it is decidable whether there is a ϕ-SSB of size at most $2^{2|\phi|}$. $\dashv$

A wide range of modal satisfiability problems can be studied in using quasi-models and mosaics. Indeed, such methods are not only useful for establishing decidability results, they can be used to obtain complexity results as well; see the Notes for further references.

Exercises for Section 6.4

6.4.1 Prove that $\mathbf{K}_t\mathbf{N}$ does not have the finite model property.

6.4.2 Give an example of an unsatisfiable Hintikka set. (Hint: work with the closure of $\{\Box(p \wedge q), \neg\Box p, \neg\Box q\}$.)

6.4.3 Extend our proof of the decidability of the tense logic of the natural numbers to a similarity type including the *next time* operator X. The semantics of this operator is given by

$$(\mathfrak{N}, V), n \Vdash X\phi \text{ iff } (\mathfrak{N}, V), n+1 \Vdash \phi.$$

6.4.4 Let F_2 be the class of frames for the basic modal similarity type in which every point has exactly two successors. Use a mosaic argument to prove that this class has a decidable satisfiability problem.

6.4.5 In this exercise we consider a version of deterministic PDL in which *every* program is interpreted as a partial function — at least, in the intended semantics. The syntax of this language is given by

$$\begin{array}{lll} \phi & ::= & p \mid \bot \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle\pi\rangle\phi \quad (p \text{ a proposition letter}) \\ \pi & ::= & a \mid \pi_1 ; \pi_2 \mid \mathrm{if}(\phi, \pi_1, \pi_2) \mid \mathrm{repeat}(\pi, \phi) \quad (a \text{ an atomic program}). \end{array}$$

In a regular model $\mathfrak{M}$ for this language, each relation R_a is a partial function, and the interpretation of the composed programs is given in the obvious way. That is, $R_{\pi_1;\pi_2}$ is the relational composition of R_{π_1} and R_{π_1}; $R_{if(\phi,\pi_1,\pi_2)}st$ holds if either $\mathfrak{M}, s \Vdash \phi$ and $R_{\pi_1}st$ or else $\mathfrak{M}, s \not\Vdash \phi$ and $R_{\pi_2}st$. Finally, we have $R_{repeat(\pi,\phi)}st$ if there is a path $sR_\pi t_1 R_\pi t_2 \ldots R_\pi t_n = t$ from s to t such that $n \geq 1$ and $t = t_n$ is the *first* t_i where ϕ holds.

(a) Prove that in a regular model, each program π is interpreted as a partial function.
(b) Prove that the class of regular models has a decidable theory over this language. (Hint: use mosaics (bricks) of the form (Φ, Λ, Π) where Φ and Λ are Hintikka sets and Π is a set of programs closed under some natural conditions.)

6.5 Undecidability via Tiling

There are lots of undecidable modal logics; indeed, even uncountably many with the polysize model property (see Exercise 6.2.4). Moreover, there are undecidable modal logics which in many other ways are rather well-behaved (we saw an example in Exercise 6.2.5). Nice as they are, these examples do not really make clear just how easily undecidable modal logics can arise, nor how serious the undecidability can be. This is especially relevant if we are working with the richer modal languages (such as PDL) typically used in computer science and other applications, and the first goal of this section is to show that natural (and on the face of it, straightforward) ideas can transform simple decidable logics into undecidable (or even *highly* undecidable) systems. While the examples are interesting in their own right, this section has a second goal: to introduce the concept of *tiling problems*.

Given a modal satisfiability problem S, to prove that S is undecidable we must reduce some known undecidable problem U to S. But which problems are the interesting candidates for reduction? Unsurprisingly, there is no single best answer to this question. As with decidability proofs, proving undecidability is something of an art: it can be very difficult, and there is no substitute for genuine insight into the satisfiability problem. Certain problems lend themselves rather naturally to modal logic, and tiling problems are a particularly nice example.

What is a tiling problem? In essence, a jigsaw puzzle. A *tile* T is simply a 1×1 square, fixed in orientation, each side of which has a *color*. We refer to these four colors as $right(T)$, $left(T)$, $up(T)$, and $down(T)$. Figure 6.3 depicts an example. (We have used different types of shading to represent the different colors.)

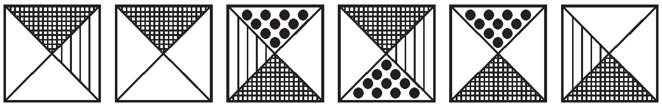


Fig. 6.3. Six distinct tile types.

Six tiles are shown in Figure 6.3. Note that if we rotated the third tile 180 degrees clockwise, it would look just like the fourth tile, and that if we rotated the first tile 180 degrees clockwise it would look just like the sixth tile. We ignore such similarities. (This is what we meant when we said that tiles are 'fixed in orientation.') That is, the diagram shows six distinct types of tile.

Now for a simple tiling problem:

> Is it possible to arrange tiles of the type just shown on a 2×4 grid in such a way that adjacent tiles have the same color on the common side?

A little experimentation shows that this *is* possible. A solution is given in Figure 6.4.

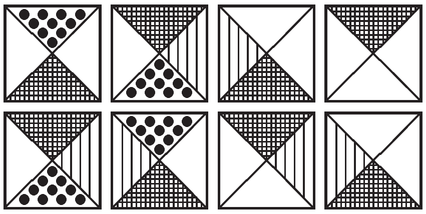


Fig. 6.4. A 2×4 tiling.

This simple idea of pattern-matching underlying tiling problems gives rise to a family of problems which can be used to analyze computational complexity and demonstrate undecidability. This is the general form that tiling problems take:

> Given a finite set of tile types $\mathcal{T}$, can we cover a certain part of $\mathbb{Z} \times \mathbb{Z}$ in such a way that adjacent tiles have the same color on the common edge? (Below, covering a grid with tiles so that adjacent colors match will be called 'tiling'.)

Some tiling problems impose additional constraints on what counts as a successful tiling (we will shortly see an example) and some are formulated as games to be played between two players (we will see an example at the end of this chapter).

To spell this out somewhat, we might describe our previous example as an instance of the 2×4 tiling problem. That is, we were given a finite set of tile types (six, to be precise), asked to tile a 2×4 grid, and no further constraints were imposed. In the remainder of this section, we are going to make use of two much harder tiling problems. The first is the:

$\mathbb{N} \times \mathbb{N}$ **tiling problem**. *Given a finite set of tile types $\mathcal{T}$, can $\mathcal{T}$ tile $\mathbb{N} \times \mathbb{N}$?*

Here is a simple instance of this problem: can we tile $\mathbb{N} \times \mathbb{N}$ using the six tile types shown? Of course! We need simply 'slot-together' copies of our solution to the 2×4 problem.

In general, however, the $\mathbb{N} \times \mathbb{N}$ tiling problem is hard, and in fact it is known to be *undecidable*. Indeed, this problem is Π^0_1-complete; that is, it is a paradigmatic example of 'ordinary undecidability.' (See Section C for further discussion of degrees of undecidability.) We won't prove this result here — see the Notes for references — but it is really quite straightforward: think of each row of tiles as encoding Turing machine tapes and states, and the matching process as governing the state transitions.

The second problem we will use is the:

$\mathbb{N} \times \mathbb{N}$ **recurrent tiling problem**. *Given a finite set of tile types $\mathcal{T}$, which includes some distinguished tile type T_1, can $\mathcal{T}$ tile $\mathbb{N} \times \mathbb{N}$ in such a way that T_1 occurs infinitely often in the first row?*

As an easy example, note that our previous six tile types recurrently tile $\mathbb{N} \times \mathbb{N}$ when either the first, the third, the fourth, or the sixth tile type is distinguished.

Now our new problem is just the $\mathbb{N} \times \mathbb{N}$ tiling problem with an additional constraint imposed — but what a difference this constraint makes! Not only is this problem undecidable, it is Σ^1_1-complete (again, see Section C).

We will prove two modal undecidability results with the aid of these problems. Both examples are based around a natural variant of Deterministic Propositional Dynamic Logic, with intersection replacing choice and iteration as program constructors; we call this variant KR. We obtain our undecidability results as follows. First we enrich KR with the *global modality*. As we will show, the combination of the intersection construct with the global modality is a powerful one: it is possible to give an extremely straightforward reduction of the $\mathbb{N} \times \mathbb{N}$ tiling problem. We then enrich KR with a modality called the *master modality*. This is also a natural operator — indeed, perhaps more natural than the global modality. As a very easy reduction from the $\mathbb{N} \times \mathbb{N}$ recurrent tiling problem reveals, the resulting system is highly undecidable.

Intersection and the global modality

Our first example vividly illustrates how easily undecidability can arise. We are going to mix two simple ingredients together, both of which are decidable, and show that the result has an undecidable satisfiability problem.

The first ingredient is a variant of DPDL, with intersection replacing choice and iteration as program constructors. Recall from Example 1.15 that DPDL is simply PDL interpreted over deterministic PDL structures (that is, PDL structures in which the relations R_a corresponding to atomic programs a are partial functions). Further, recall from Example 1.26 that modalities built with the intersection constructor (that is, modalities of the form $\langle \pi_1 \cap \pi_2 \rangle$) are interpreted by the relation $R_{\pi_1} \cap R_{\pi_2}$, where R_{π_1} is the relation corresponding to $\langle \pi_1 \rangle$ and R_{π_2} the relation corresponding to $\langle \pi_2 \rangle$.

In what follows we will not use the entire language; instead we will work with a fragment (called KR) which consists of all formulas without occurrences of $*$ and $\cup$. That is, KR contains precisely the following formulas ϕ:

$$\begin{array}{lll} \phi & ::= & p \mid \bot \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle \pi \rangle \phi \quad (p \text{ a proposition letter}) \\ \pi & ::= & a \mid \pi_1 \,;\, \pi_2 \mid \pi_1 \cap \pi_2 \quad (a \text{ an atomic program}). \end{array}$$

The KR language is rather simple: essentially it allows us to state whether or not different sequences of (deterministic) programs terminate in the same state when executed in parallel. Note that (over deterministic PDL structures) a selection argument immediately shows that it is decidable. Over deterministic structures, *every* modal operator in KR is interpreted by a partial function. (This is because all atomic programs are modeled by partial functions, and the only program constructors we have at our disposal are composition and intersection.) It follows that if a sentence ϕ from KR is satisfiable in a deterministic model, then it is satisfiable in a *finite* deterministic model; the proof is essentially the same as that of Corollary 6.9.

The second ingredient is even simpler. We are going to add the *global modality* A to our fragment. This is an interesting operator that we are going to discuss in detail in Section 7.1; for present purposes we only need to know two things about it. First, it is interpreted as follows: $\mathfrak{M}, w \Vdash A\phi$ if for all v in $\mathfrak{M}$ we have $\mathfrak{M}, v \Vdash \phi$. Thus, as its name suggests, the global modality is a modal operator which allows us to express global facts. Second, A has a decidable satisfiability problem. (To see this, simply observe that A is an **S5** operator, and we know that **S5** is decidable.) Thus, on its own, A is pretty harmless.

But what happens when we add A to KR? The resulting language called KRA can talk about computations in a very natural (and very powerful) way. For example,

$$A(\langle a \rangle \top \to \psi)$$

expresses that in every state of a computation, ψ is a precondition for the program

a to have a terminating execution. As we will now show, KRA has crossed the border into undecidability.

Theorem 6.31 *Assume that the language has at least two atomic programs. Then the satisfiability problem for* KRA *is undecidable. To be precise, it is* Π^0_1*-hard.*

Proof. We show this by reducing the $\mathbb{N}\times\mathbb{N}$ tiling problem to the KRA satisfiability problem; the undecidability (and Π^0_1-hardness) of the satisfiability problem will follow from the known undecidability (Π^0_1-hardness) of the $\mathbb{N}\times\mathbb{N}$ tiling problem.

Recall that the $\mathbb{N}\times\mathbb{N}$ tiling problem asks: given a finite set of tile types $\mathcal{T}$, can $\mathcal{T}$ tile $\mathbb{N}\times\mathbb{N}$? Putting this more formally: does there exist a function $t : \mathbb{N}\times\mathbb{N} \to \mathcal{T}$ such that

$$\begin{aligned} right(t(n,m)) &= left(t(n+1,m)) \\ up(t(n,m)) &= down(t(n,m+1))? \end{aligned}$$

We will reduce $\mathbb{N}\times\mathbb{N}$ tiling to the satisfiability problem as follows. Let $\mathcal{T} = \{T_1,\ldots,T_k\}$ be the given set of tile types. We will construct a formula $\phi_{\mathcal{T}}$ such that

$$\mathcal{T} \text{ tiles } \mathbb{N}\times\mathbb{N} \text{ iff } \phi_{\mathcal{T}} \text{ is satisfiable.} \tag{6.8}$$

If we succeed in constructing such a formula it follows that the KRA-satisfiability problem is undecidable. (For suppose it was decidable. Then we could solve the $\mathbb{N}\times\mathbb{N}$ tiling problem as follows: given $\mathcal{T}$, form $\phi_{\mathcal{T}}$, and use the putative KRA-satisfiability algorithm to check for satisfiability. By (6.8) this would solve the tiling problem — which is impossible.)

The construction of $\phi_{\mathcal{T}}$ proceeds in three steps. First, we show how to use KRA to demand 'gridlike' models. Second, we show how to use KRA to demand that a tiling exists on this 'grid.' Finally we prove (6.8).

Step 1. Forcing the grid. The basic idea is to let the nodes in $\mathfrak{M}$ mimic the nodes in $\mathbb{N}\times\mathbb{N}$, and to use two relations R_r and R_u to mimic the 'to-the-right' and the 'up' functions of $\mathbb{N}\times\mathbb{N}$. To get the gridlike model we want, we simply demand that R_r and R_u commute:

$$\phi_{grid} := A\langle (r\,;u) \cap (u\,;r)\rangle\top.$$

This says that everywhere in the model it is possible to make a 'to-the-right transition followed by an up transition' and an 'up transition followed by a to-the-right transition,' and both these transition sequences lead to the same point. (Note that this is all we need to say, since by assumption R_r and R_u are partial functions.)

Step 2. Tiling the model. We will 'tile the model' by making use of proposition letters $t_1,\ldots,t_k$ which correspond to the tile types in $\mathcal{T}$. The basic idea is simple: we want t_i to be true at a node w iff a tile of type T_i is placed on w. Of course, not any placement of tiles will do: we want a genuine tiling. But the following three demands ensure this:

(i) Exactly one tile is placed at each node:

$$\phi_1 := A\left(\bigvee_{i=1}^{k} t_i \wedge \bigwedge_{1\le i<j\le k} \neg(t_i \wedge t_j)\right).$$

(ii) Colors match going right:

$$\phi_2 := A\left(\bigvee_{right(T_i)=left(T_j)} (t_i \wedge \langle r\rangle t_j)\right).$$

(iii) Colors match going up:

$$\phi_3 := A\left(\bigvee_{up(T_i)=down(T_j)} (t_i \wedge \langle u\rangle t_j)\right).$$

Putting this together, we define $\phi_{\mathcal{T}} := \phi_{grid} \wedge \phi_1 \wedge \phi_2 \wedge \phi_3$.

Step 3. Proving the equivalence. We now show that (6.8) holds. Assume first that $t : \mathbb{N}\times\mathbb{N} \to \mathcal{T}$ is a tiling of $\mathbb{N}\times\mathbb{N}$. Construct a satisfying model for $\phi_{\mathcal{T}}$ as follows.

$$\begin{aligned} W &= \{w_{n,m} \mid n,m \in \mathbb{N}\} \\ R_r &= \{(w_{n,m}, w_{n+1,m}) \mid n,m \in \mathbb{N}\} \\ R_u &= \{(w_{n,m}, w_{n,m+1}) \mid n,m \in \mathbb{N}\} \\ V(t_i) &= \{w_{n,m} \mid n,m \in \mathbb{N} \text{ and } t(n,m) = T_i\}. \end{aligned}$$

Clearly, $\phi_{\mathcal{T}}$ holds at any state w of $\mathfrak{M}$.

For the converse, let $\mathfrak{M}$ be a model such that $\mathfrak{M}, w_0 \Vdash \phi_{\mathcal{T}}$. It follows from $\mathfrak{M}, w_0 \Vdash \phi_{grid}$ that there exists a function $f : \mathbb{N}\times\mathbb{N} \to W$ such that $f(0,0) = w_0$, $R_r f(n,m) f(n+1,m)$ and $R_u f(n,m) f(n,m+1)$. Define the tiling $t : \mathbb{N}\times\mathbb{N} \to \mathcal{T}$ by

$$t(n,m) = T_i \text{ iff } \mathfrak{M}, f(n,m) \Vdash t_i.$$

By ϕ_1, t is well-defined and total. Moreover, if $t(n,m) = T_i$ and $t(n+1,m) = T_j$, then $R_r f(n,m) f(n+1,m)$, and both $\mathfrak{M}, f(n,m) \Vdash t_i$ and $\mathfrak{M}, f(n+1,m) \Vdash t_j$. Given that w_0 satisfies ϕ_2, we conclude that $right(T_i) = left(T_j)$. Similarly, because of ϕ_3, if $t(n,m) = T_i$ and $t(n,m+1) = T_j$, then $up(T_i) = down(T_j)$. Thus, $\mathcal{T}$ tiles $\mathbb{N}\times\mathbb{N}$. $\dashv$

The above proof clearly depends on having two deterministic atomic programs at our disposal. But what happens if we only have one? It should be clear that then

$\cap$ cannot do any interesting work for us, and in fact the language has a *decidable* satisfiability problem; see Exercise 6.5.1.

We now know that KRA-satisfiability is undecidable (given more than one atomic program) but *how undecidable* is it? In particular can we also prove a Π_1^0 upper bound to match the Π_1^0-hardness result? (That is, can we show that we are dealing with a case of 'ordinary undecidability'?) To prove this, it suffices to show that the validities of KRA form an r.e. set. Now we could do this by devising a recursive axiomatization of the KRA-validities, but by making use of a general lemma from correspondence theory we can establish the result more straightforwardly.

Lemma 6.32 *If* K *is a class of frames defined by a first-order formula, then its modal logic is recursively enumerable.*

Proof. Assume that the first-order formula α defines K, where α is built using only relation symbols of arity 2 or higher, and identity. Then, a modal formula ϕ is valid on K iff it is valid on all frames in K iff

$$\alpha \models \forall x \forall P_1 \ldots \forall P_n \, ST(\phi), \tag{6.9}$$

where $P_1, \ldots, P_n$ are unary predicate symbols corresponding to the proposition letters in ϕ. As the predicate variables $P_1, \ldots, P_n$ do not occur in α, (6.9) is equivalent to $\alpha \models \forall x \, ST(\phi)$. But this is an ordinary first-order implication, which is an r.e. notion. Hence, modal validity on K is an r.e. notion as well. $\dashv$

Theorem 6.33 *Assume that our language has at least two, but at most finitely many atomic programs. Then the satisfiability problem for* KRA *is* Π_1^0*-complete.*

Proof. The Π_1^0 lower bound is given by the encoding of the $\mathbb{N} \times \mathbb{N}$ tiling problem in the proof of Theorem 6.31. For the Π_1^0 upper bound we show that the validity problem for KRA is r.e. The standard translations for the constructors ; and $\cap$ are given in Section 2.4; both are first-order. (Recall that the $*$ constructor is the only part of PDL that takes us out of first-order logic.) The standard translation for A is obvious (and clearly first-order):

$$ST(A\phi) = \forall y \, [y/x] ST(\phi).$$

Thus — assuming we are working with a language of KRA that contains at most finitely many atomic programs — the required class of frames is defined by

$$\bigwedge_{\alpha \text{ atomic}} \forall xyz \, (R_\alpha xy \wedge R_\alpha xz \to y = z).$$

Hence, by Lemma 6.32, the modal logic of the class of frames for KRA is r.e. as required. $\dashv$

Intersection and the master modality

Our next example illustrates how easily *high* undecidability can arise. Once again, we will enrich the KR language, but this time with the *master modality*. As we will see, the resulting language KR$\boxplus$ has a Σ_1^1-complete satisfiability problem.

Like the global modality, the master modality $\boxplus$ is a tool for expressing general constraints in the object language, but it works rather differently. A formula of the form $\boxplus\phi$ is true at a node w iff ϕ is true at all nodes reachable by any finite sequence of atomic transitions from w. Formally,

$$w \Vdash \boxplus\phi \text{ iff } v \Vdash \phi \text{ for all } v \text{ such that } (w,v) \in \left(\bigcup_{a \text{ atomic}} R_a\right)^* .$$

That is, $\boxplus$ explores the reflexive transitive closure of the union of all the relations used to interpret the atomic programs. If we only have finitely many atomic programs $a_1, \ldots, a_n$, the master modality is simply shorthand for the PDL modality $[(a_1 \cup \cdots \cup a_n)^*]$. From a computational perspective, this modality is arguably even more natural than the global modality: it is a way of looking at what must happen throughout the space of possible computations. (It has other natural interpretations as well. For example, if we interpret our basic modalities as in multi-agent epistemic logic — that is, $[a]\phi$ means 'agent a knows that ϕ' — then $\boxplus$ is the 'common knowledge' operator.)

But, for all its naturalness, the master modality can be extremely dangerous. Let us see what happens when we add it to KR. First, observe that KR$\boxplus$ must be undecidable. (There is nothing new to prove here; simply observe that if we systematically replace every occurrence of A in the proof of Theorem 6.31 by $\boxplus$, the argument still goes through.) But can we prove a matching Π_1^0 upper bound? We certainly cannot appeal to Lemma 6.32; while the global modality was essentially first-order, the master modality is not. (As with the $*$ constructor of PDL, its natural correspondence language is infinitary; see Section 2.4.) And indeed, any attempt to recursively enumerate the validities of KR$\boxplus$ is bound to fail.

Theorem 6.34 *The satisfiability problem for* KR$\boxplus$ *is highly undecidable. To be precise, it is* Σ_1^1*-hard.*

Proof. We show this by reducing the recurrent tiling problem to the KR$\boxplus$-satisfiability problem; the Σ_1^1-hardness of the satisfiability problem will follow from the known Σ_1^1-hardness of the recurrent tiling problem.

Recall that the recurrent tiling problem asks: given a finite set of tile types $\mathcal{T}$, which includes some distinguished tile type T_1, can $\mathcal{T}$ tile $\mathbb{N}\times\mathbb{N}$ in such a way that T_1 occurs infinitely often in the first row? Putting this more formally: does there exist a function $t : \mathbb{N}\times\mathbb{N} \to \mathcal{T}$ such that

$$right(t(n,m)) \quad = \quad left(t(n+1,m))$$

$$\begin{array}{rcl} up(t(n,m)) & = & down(t(n,m+1)) \\ \{n \mid t(n,0)=T_1\} & & \text{is infinite?} \end{array}$$

We reduce $\mathbb{N} \times \mathbb{N}$ recurrent tiling to KR$\boxplus$-satisfiability as follows. Let $\mathcal{T} = \{T_1, \ldots, T_k\}$ be the set of tile types. We will define a formula $\phi_{\mathcal{T},T_1}$ such that

$$\mathcal{T} \text{ and } T_1 \text{ recurrently tile } \mathbb{N} \times \mathbb{N} \text{ iff } \phi_{\mathcal{T},T_1} \text{ is satisfiable.} \tag{6.10}$$

Most of the real work was done in the proof of Theorem 6.31. Let us simply take the earlier encoding $\phi_{\mathcal{T}}$ and replace every occurrence of A with $\boxplus$. Call the result $\phi^*_{\mathcal{T}}$. This formula reduces the $\mathbb{N} \times \mathbb{N}$ tiling problem to the KR$\boxplus$-satisfiability problem.

To reduce the *recurrent* tiling problem, it remains to ensure that our distinguished tile T_1 occurs infinitely often on the first row. As t_1 is the proposition letter corresponding to T_1, this means we want to force t_1 to be true at nodes of the form $t(n,0)$ for infinitely many n. To do this, we will introduce a new proposition letter *first-row* and then define:

$$\phi_{rec} := \textit{first-row} \wedge \boxplus[u]\neg\textit{first-row} \wedge \boxplus(\textit{first-row} \rightarrow \langle r\rangle\diamondplus(\textit{first-row} \wedge t_1)).$$

Suppose that ϕ_{rec} is satisfied at some point w_0 of a grid-like model. It follows that *first-row* is satisfied at w_0; that *first-row* can only be satisfied at points reachable by a finite number of R_r transitions from w_0; and that for infinitely many distinct natural numbers n, $w_0 \Vdash \langle r\rangle^n(\textit{first-row} \wedge t_1)$.

So, let $\phi_{\mathcal{T},T_1}$ be the conjunction of $\phi^*_{\mathcal{T}}$ and ϕ_{rec}. Then (6.10) holds. $\dashv$

To conclude this section, two general remarks. First, the examples in this section were clearly chosen to make the undecidability proofs run as smoothly as possible. In particular, our examples hinged on the use of $\cap$ to force the existence of the grid. What happens if we are working in languages without this constructor? That is, how widely applicable is this method for proving undecidability?

Suppose we are working with an arbitrary modal language, and we want to establish the undecidability of its satisfiability problem. If we abstract from the proof of Theorem 6.31, we see that there is one ingredient that will always be needed to make similar arguments go through: sufficient 'global' expressive power. This power may arise directly through the presence of additional operators, or it may arise indirectly through special features of the class of models under consideration, but one way or another we will need it. On the other hand, we do not need the $\cap$ constructor; Exercise 6.5.2 is a nice example.

Second, we have discussed tiling problems as if they were useful only for establishing different grades of undecidability. In fact, they can also be used to analyze the complexity of *decidable* problems: for example, there are NP-hard, PSPACE-hard, and EXPTIME-hard tiling problems (see the Notes for further references).

At the end of this chapter we will use a 2-player tiling problem to show that the satisfiability problem for **PDL** is EXPTIME-hard.

Exercises for Section 6.5

6.5.1 Show that KRA-satisfiability is decidable if we have only one atomic program at our disposal. (This result can be proved via a finite model property argument.)

6.5.2 (i) Show that the satisfiability problem of the following 'tiling' logic $\mathbf{Tile}_1$ is undecidable. $\mathbf{Tile}_1$ is a normal modal logic with three diamonds $\langle u \rangle$, $\langle r \rangle$ and $\Diamond$, defined by the following (Sahlqvist) axioms:

$$\langle u \rangle p \to [u]p \text{ and } \langle r \rangle p \to [r]p \tag{6.11}$$
$$\langle r \rangle \langle u \rangle p \to [u] \langle r \rangle p$$
$$\Diamond \Diamond p \to \Diamond p$$
$$\langle u \rangle p \to \Diamond p \text{ and } \langle r \rangle p \to \Diamond p.$$

(ii) Now use this logic plus the standard translation to conclude that the three variable fragment of first-order logic (without function symbols, but possibly with equality) is undecidable.

(iii) Let $\mathbf{Tile}_2$ be obtained from $\mathbf{Tile}_1$ by omitting axiom (6.11). Show that $\mathbf{Tile}_2$ is still undecidable. (Hint: Reduce the satisfiability problem of $\mathbf{Tile}_1$ to that of $\mathbf{Tile}_2$.)

(iv) Conclude that first-order logic with three variables, but without equality is undecidable.

(v) Use a similar tiling logic to show that first-order logic with one variable, two unary function symbols, and only unary predicate symbols is undecidable. (Hint: adjust the standard translation so that it exploits the unary function symbols directly.)

6.6 NP

The interpretation method (and in particular, interpretations in $\mathrm{S}n\mathrm{S}$) is a powerful and widely applicable way of proving decidability. Nevertheless, it has disadvantages. Reducing the satisfiability problems of what are often rather simple modal logics to $\mathrm{S}n\mathrm{S}$ is using a sledgehammer to crack a nut. The decision problem for $\mathrm{S}n\mathrm{S}$ is non-elementary. This means that the time required to decide whether an arbitrary formula ϕ is decidable cannot be bounded by any finite tower of exponentials of the form

$$2^{2^{\cdot^{\cdot^{\cdot^{2^{|\phi|}}}}}}.$$

The use of filtrations to establish decidability is open to similar objections. A filtration is typically $2^{|\phi|}$ in the size of the input formula. But it is not feasible to enumerate all the models up to this size even for quite small values of $|\phi|$. And even a nondeterministic Turing machine, which could 'guess' a filtration in one move (see Appendix C and the discussion below), would still be faced with the immensely costly task of checking that ϕ was true on this huge structure (to use the terminology discussed in Appendix C, filtrations typically offer us NEXPTIME algorithms). Indeed, of the three decidability techniques discussed so far, only the mosaic method (which 'deconstructs' models locally) respects what is special about modal logic; and as we will learn in Section 7.4, the mosaic method can be used to give essentially optimal satisfaction algorithms.

But this is jumping ahead. In this section and the three that follow, we will use concepts drawn from *computational complexity theory* to present a more fine-grained analysis of modal satisfiability. This analysis is interesting for two reasons. First, by making use of only three central complexity classes (NP, PSPACE and EXPTIME), we will be able to present a classification of modal satisfiability that covers many important logics. Secondly, in many cases the techniques involved have a distinctly modal flavor: essentially, the work boils down to a refined analysis of the finite model property.

We begin our analysis with the class NP, the class of problems solvable using nondeterministic polynomial time algorithms. We first review the central ideas underlying this complexity class and their import for modal satisfiability problems. Then, using examples from multi-modal and tense logic, we show how simple selection arguments can be used to prove NP-completeness results. Finally, we apply the same method to prove a more general result: every normal modal logic extending **S4.3** has an NP-complete satisfiability problem.

When a problem $\mathcal{P}$ is said to be complete with respect to a complexity class C, two things are being claimed. The first is that $\mathcal{P}$ belongs to C; that is, there is an algorithm using only the resources permitted by C that solves $\mathcal{P}$. For example, if $\mathrm{C} = \mathrm{NP}$ this means that there exists a non-deterministic polynomial time algorithm for solving $\mathcal{P}$. The second claim is that $\mathcal{P}$ is C-hard; that is, any other problem in C is polynomial time reducible to $\mathcal{P}$.

Now, as far as the satisfiability problem for normal modal logics is concerned, NP-hardness is a triviality: all (consistent) normal modal logics have NP-hard satisfiability problems. The point is this. The classic NP-hard problem is the satisfiability problem for propositional logic. But as every normal modal logic is an extension of propositional logic, every (consistent) normal modal logic has a satisfiability problem at least as hard as that for propositional logic. Thus — for the class NP — our work is somewhat simplified: we are simply looking for normal modal logics whose satisfiability problem belongs to NP.

What sort of problems belong to NP? Many problems decompose naturally into the following two steps: a *search for a solution* followed by a *verification of the solution*. In general, search is expensive, but by thinking in terms of non-deterministic algorithms we can abstract away from this expense: if a solution exists, such an algorithm will find it in one non-deterministic step. (If necessary, consult Section C for further discussion.) This abstraction leaves us free to concentrate on the verification step, and leads us to isolate the class NP: a problem belongs to NP iff it has the above general profile (that is, a non-deterministic choice of a solution followed by a verification) *and moreover* the verification step is tractable (that is, solvable in polynomial time).

How do such ideas bear on modal satisfiability? The key idea we need is embodied in the following lemma.

Lemma 6.35 *Let τ be a finite similarity type. Let Λ be a consistent normal modal logic over τ with the polysize model property with respect to some class of models* M. *If the problem of deciding whether $\mathfrak{M} \in \mathsf{M}$ is computable in time polynomial in $|\mathfrak{M}|$, then Λ has an NP-complete satisfiability problem.*

Proof. As noted above, the NP-hardness of the problem is immediate, so it remains to prove the existence of an algorithm in NP that solves Λ-satisfiability. Given ϕ, non-deterministically choose a model $\mathfrak{M}$ whose size is polynomial in the size of ϕ. Because $\mathfrak{M}$ is polysize in $|\phi|$, we can check in time polynomial in $|\phi|$ whether $\mathfrak{M}$ verifies ϕ. For the special case of the basic modal language, this may be seen as follows.

Let $||\mathfrak{M}||$ denote the sum of the number states in $\mathfrak{M}$ and the number of pairs in $\mathfrak{M}$'s binary relation $R^{\mathfrak{M}}$. Let $\psi_1, \ldots, \psi_k$ be an enumeration of the subformulas of ϕ, in increasing length. So $\psi_k = \phi$ and if ψ_i is a subformula of ψ_j, then $i < j$. Notice that $k \leq |\phi|$. One can show by induction on m that we can mark each state w in $\mathfrak{M}$ with ψ_j or $\neg\psi_j$, for $j = 1, \ldots, m$, depending on whether or not ψ_j is true at w in time $\mathcal{O}m \cdot ||\mathfrak{M}||$. The only non-trivial case is if $\psi_{m+1} = \Diamond\psi_j$, for some $j < m + 1$. But in that case we mark w with $\Box\psi_j$ if some v with Rwv is marked with ψ_j. By our induction hypothesis, every state is already marked with ψ_j or $\neg\psi_j$, this step can be carried out in time $\mathcal{O}||\mathfrak{M}||$. Since $\mathfrak{M}$ is polysize in $|\phi$, so is $||\mathfrak{M}||$. Hence, checking whether $\mathfrak{M}$ satisfies ϕ can indeed be done in time polynomial in $|\phi|$.

Finally, then, because membership in M is decidable in time polynomial in $|\mathfrak{M}|$, and $|\mathfrak{M}|$ is polynomial in $|\phi|$, we can check in time polynomial in $|\phi|$ that $\mathfrak{M}$ is in M. $\dashv$

Where did we use the assumption that τ is a finite similarity type in the proof of Lemma 6.35? Essentially, it allows us to check whether $\mathfrak{M}$ verifies ϕ in time polynomial in ϕ and in $|\mathfrak{M}|$. The key point is this: when working with a fixed finite similarity type, we are actually working within a finite-variable fragment, say with l variables. This allows us to restrict our attention to only finitely many relations of arity at most l in $\mathfrak{M}$. While the total number of tuples in all relations in $\mathfrak{M}$ may be huge, it is nonetheless independent of ϕ; see Exercise 6.6.2 for further elaborations.

Note that the second demand — that M-membership be polynomial time decidable — is vital. As the reader was asked to show in Exercise 6.2.4, the polysize model property alone is insufficient to ensure decidability, let alone the existence of a solution in NP. However, for many important logics this property can be established by appealing to the following standard result.

Lemma 6.36 *If* F *is a class of frames definable by a first-order sentence, then the problem of deciding whether* $\mathfrak{F}$ *belongs to* F *is decidable in time polynomial in the size of* $\mathfrak{F}$.

Proof. Left as Exercise 6.6.1. $\dashv$

We will show that many normal modal logics are NP-complete. The proofs revolve around one central idea: the construction of polysize models by the selection of polynomially many points from some given satisfying model.

For our first example, we return to the multi-modal language containing n unary modal operators discussed earlier (see Corollary 6.9). Recall that F_1^n is the class of frames for this language in which each relation is a partial function, M_1^n is the class of models built over F_1^n, and $\mathbf{K}_n\mathbf{Alt}_1$ is its logic.

Theorem 6.37 $\mathbf{K}_n\mathbf{Alt}_1$ *has an NP-complete satisfiability problem.*

Proof. We already showed that this logic has the strong f.m.p., but the selection argument we used generated models exponential in size of the input formula. A simple refinement of the method shows that $\mathbf{K}_n\mathbf{Alt}_1$ actually has the *polysize* model property.

Given a formula ϕ of this language and a model $\mathfrak{M} = (W, R, V)$ we define a selection function s as follows:

$$\begin{aligned} s(p, w) &= \{w\} \\ s(\neg\phi, w) &= s(\phi, w) \\ s(\phi \wedge \psi, w) &= s(\phi, w) \cup s(\psi, w) \\ s(\langle a \rangle \psi, w) &= \{w\} \cup \bigcup_{\{w' \mid R_a w w'\}} s(\psi, w') \end{aligned}$$

Intuitively, $s(\phi, w)$ selects the nodes actually needed when evaluating ϕ in $\mathfrak{M}$ at w — and indeed, it follows by induction on the structure of ϕ that for all nodes w of $\mathfrak{M}$, and all formulas ϕ

$$\mathfrak{M}, w \Vdash \phi \text{ iff } \mathfrak{M} \upharpoonright s(\phi, w), w \Vdash \phi.$$

It is clear that $\mathfrak{M} \upharpoonright s(\phi, w) \in \mathsf{M}_1^n$. So let us look at size of the new model. If $\mathfrak{M} \in \mathsf{M}_1^n$, we claim that $|s(\phi, w)| \leq |\phi| + 1$. To see this, note that only occurrences of modalities in ϕ cause new nodes to be adjoined to $s(\phi, w)$. This adjunction of points is carried out in the fourth clause of the inductive definition for s, which tells us to adjoin every state w' such that $R_a w w'$. Because $\mathfrak{M} \in \mathsf{M}_1^n$, every relation R_a

is a partial function; hence if such a w' exists, it is unique. In short, $\mathbf{K}_n\mathbf{Alt}_1$ has the polysize model property: simply counting the number of occurrences of modal operators in ϕ and adding one gives us an upper bound on the size of the domain of the required satisfying model.

By Lemma 6.36, membership in $\mathbf{K}_n\mathbf{Alt}_1$ is decidable in polynomial time, for this is a class of frames definable by a first-order sentence — namely the conjunction of sentences that say that each of the n relations is a partial function.

The result follows by Lemma 6.35. $\dashv$

The argument for $\mathbf{K}_n\mathbf{Alt}_1$ shows the selection method in its simplest form: given *any* model for ϕ we build a new polysize model for ϕ by making a suitable selection of polynomially many points. This simple form of argumentation is applicable to a number of logics, a particularly noteworthy example being **S5**. Given any **S5** model for ϕ, it is possible to select $m + 1$ points from this model (where m is the number of modality occurrences in ϕ) which suffice to construct a new **S5** model for ϕ, and the NP-completeness of **S5** follows straightforwardly. We leave the details as Exercise 6.6.4 and turn our attention to a modification of the point selection method frequently needed in practice: a *detour via finite models*.

Both $\mathbf{K}_n\mathbf{Alt}_1$ and **S5** are very simple logics; in neither case is it difficult to determine which points should be selected. In other cases, we may not be so fortunate. Suppose we are trying to show that a logic Λ has the polysize model property, and we already know that Λ has the f.m.p. Then, instead of trying to select points from an arbitrary model, we are free to select points from a finite model, or even a point-generated submodel of a finite model. This often gives us an easy way of zooming in on the crucial points. In particular, when we are working with models based on finite orderings it makes sense to talk of choosing points that are maximal (or minimal) in the frame ordering that satisfy some subformula; such extremal points are often the vital ones. As an example of such an argument, let us consider $\mathbf{K}_t\mathbf{4.3}$, the temporal logic of linear frames (in the basic temporal language).

Theorem 6.38 $\mathbf{K}_t\mathbf{4.3}$ *has an NP-complete satisfiability problem.*

Proof. We will first show that $\mathbf{K}_t\mathbf{4.3}$ has the polysize model property. Let ϕ be a formula of the basic temporal language that is satisfiable on a $\mathbf{K}_t\mathbf{4.3}$ model. As $\mathbf{K}_t\mathbf{4.3}$ has the f.m.p. with respect to the class of weak total orders (see Definition 4.37 and Corollary 6.8), there is a finite weakly totally ordered model $\mathfrak{M} = (T, \leq, V)$ containing a node t such that $\mathfrak{M}, t \Vdash \phi$. We now build a polysized model for ϕ by selecting points from $\mathfrak{M}$.

Let $F\psi_1, \ldots, F\psi_k$ and $P\theta_1, \ldots, P\theta_l$ be all subformulas of ϕ of the form $F\psi$ and $P\theta$, respectively, that are satisfied in $\mathfrak{M}$. For each formula $F\psi_i$ choose a point u_i such that $\mathfrak{M}, u_i \Vdash \psi_i$ and u_i is a *maximal* point in the $\leq$-ordering with this property. Similarly, for each formula $P\theta_j$ choose a point v_j satisfying θ_j that is *minimal* in the $\leq$-ordering with respect to this property. Let $\mathfrak{M}'$ $(= (T', \leq', V'))$ be $\mathfrak{M} \upharpoonright \{t, u_1, \ldots, u_k, v_1, \ldots, v_l\}$. As $\leq$ is a weak total ordering of T, $\leq'$ is a weak total ordering of T'. Furthermore, the number of nodes in $\mathfrak{M}'$ does not exceed $m + 1$, where m is the number of modalities in ϕ, thus $\mathfrak{M}'$ is a polysize model in the correct class. It remains to show that $\mathfrak{M}', t \Vdash \phi$, but this follows straightforwardly by induction on the structure of ϕ.

As the class of weak total orders is definable using a first-order sentence, the NP-completeness of $\mathbf{K}_t 4.3$ follows from Lemma 6.36 and the polysize model property that we have just established. $\dashv$

We are ready to prove a general complexity result for the basic modal language: all normal logics extending **S4.3** have an NP-complete satisfiability problem. Recall from our discussion of Bull's theorem in Section 4.9 that an **S4.3** frame is a frame that is rooted, transitive, and connected $(\forall xy\,(Rxy \vee Ryx))$; note that all such frames are reflexive. Bull's Theorem tells us that all normal modal logics extending **S4.3** have the finite frame property with respect to a class of **S4.3** frames. By making a suitable selection from models based on such frames, we can prove that every such logic has the polysize model property. Then, by using the fact that every normal logic extending **S4.3** has a negative characterization in terms of finite sets of finite frames (Theorem 4.103), we will be able to prove that all these satisfiability problems are NP-complete.

First we need the following lemma; it is really just Lemma 4.98, which linked bounded morphisms and covering lists, stated in purely modal terms.

Lemma 6.39 *Let $\mathfrak{F}$ and $\mathfrak{G}$ be two finite* **S4.3** *frames. Then the following two statements are equivalent:*

(i) *There exists a surjective bounded morphism from $\mathfrak{F}$ to $\mathfrak{G}$.*
(ii) *$\mathfrak{G}$ is isomorphic to a subframe of $\mathfrak{F}$ that contains a maximal point of $\mathfrak{F}$.*

Proof. First suppose that f is a surjective bounded morphism from $\mathfrak{F}$ to $\mathfrak{G}$. Let w_{max} be a maximal point in $\mathfrak{F}$, and let $\widehat{W}$ consist of w_{max} together with exactly one maximal world in $f^{-1}[v]$ for every point v of $\mathfrak{G}$ such that $v \neq f(w_{max})$. Then $\widehat{\mathfrak{F}} = \mathfrak{F} \upharpoonright \widehat{W}$ is the subframe we want.

Conversely, suppose that $\widehat{W}$ is a subset of the points in $\mathfrak{F}$, such that $\widehat{W}$ contains a maximal point w_{max}, and $\mathfrak{F} \upharpoonright \widehat{W}$ is isomorphic to $\mathfrak{G}$. We claim that the following defines a bounded morphism from $\mathfrak{F}$ onto $\mathfrak{F} \upharpoonright \widehat{W}$: $f(w) = w$, for $w \in \widehat{W}$; and if $w \notin \widehat{W}$, then $f(w)$ is a minimal world $\widehat{w} \in \widehat{W}$ such that $Rw\widehat{w}$ (that is, for any w', if Rww' then $R\widehat{w}w'$). Note that such a minimal world must always exist, since $w_{max} \in \widehat{W}$, thus f is well defined. (In short, f maps 'missing points' to successors that are as close as possible. We used the same idea to define the bounded morphism in the proof of Bull's Theorem.) Clearly f is surjective. So suppose

Rww'. Since $Rw'f(w')$ and R is transitive, we have $Rwf(w')$. By definition, $f(w)$ is a minimal element in $\widehat{W}$ such that $Rwf(w)$, thus $Rf(w)f(w')$ and f satisfies the forth condition on bounded morphisms. Finally, suppose $Rf(w)f(w')$. As $Rwf(w)$, by the transitivity of R we have $Rwf(w')$. Since $f(f(w')) = f(w')$, the back condition for bounded morphisms is also satisfied and we have shown that $\mathfrak{F} \restriction \widehat{W}$ is a bounded morphic image of $\mathfrak{F}$. As $\mathfrak{F} \restriction \widehat{W}$ is isomorphic to $\mathfrak{G}$, $\mathfrak{G}$ is a bounded morphic image of $\mathfrak{F}$ as well. $\dashv$

We now show that any normal modal logic extending **S4.3** has the polysize model property.

Lemma 6.40 *Let Λ be a normal modal logic such that* $\mathbf{S4.3} \subseteq \Lambda$. *Any formula ϕ that is satisfiable on a frame for Λ is satisfiable on a frame for Λ that contains at most $m + 2$ states, where m is the number of occurrences of modal operators in ϕ.*

Proof. Suppose ϕ is satisfiable on a frame for Λ. By Bull's Theorem, Λ has the finite frame property, thus there is a finite model based on a Λ-frame that satisfies ϕ at some point w_0. Let $\mathfrak{M}$ be the submodel of this model that is generated by w_0. Clearly $\mathfrak{M}, w_0 \Vdash \phi$, and as formation of generated submodels preserves modal validity, $\mathfrak{M}$ is based on a frame for Λ.

Now we select points. Let $\Diamond\psi_1, \ldots, \Diamond\psi_k$ be all the $\Diamond$-subformulas of ϕ that are satisfied at w_0. For each $1 \leq i \leq k$, select a point w_i that is maximal with respect to the property of satisfying ψ_i. These are the points needed to ensure that ϕ is satisfied in the polysize model at w_0, but if we select only w_0 and these points, we have no guarantee that we have constructed a Λ-frame. However, as we will now see, we *can* guarantee this if we glue on a maximal point. So, let w_{k+1} be such a point and define

$$\widehat{\mathfrak{M}} := \mathfrak{M} \restriction \{w_0, w_1, \ldots, w_k, w_{k+1}\}.$$

$\widehat{\mathfrak{M}}$ contains at most $m + 2$ points, where m is the number of modal operators in ϕ. Moreover, it *is* based on a Λ-frame. To see this, note that the frame underlying $\widehat{\mathfrak{M}}$ is a subframe of the frame underlying $\mathfrak{M}$ that satisfies the requirements of item (ii) of Lemma 6.39; hence there is a surjective bounded morphism from $\mathfrak{M}$ to $\widehat{\mathfrak{M}}$. Such morphisms preserve modal validity, thus as $\mathfrak{M}$ is a Λ-model, so is $\widehat{\mathfrak{M}}$.

It remains to ensure that $\widehat{\mathfrak{M}}, w_0 \Vdash \phi$. We prove by induction that for all subformulas ψ of ϕ, and all i such that $0 \leq i \leq k$, that

$$\mathfrak{M}, w_i \Vdash \psi \text{ iff } \widehat{\mathfrak{M}}, w_i \Vdash \psi.$$

The only interesting step is for formulas of the form $\Diamond\psi$. Suppose that $\mathfrak{M}, w_i \Vdash \Diamond\psi$ (thus $\psi = \psi_j$ for some $1 \leq j \leq k$). Since $\mathfrak{M}$ is point-generated by w_0 and transitive, it follows that Rw_0w_i, hence $\mathfrak{M}, w_0 \Vdash \Diamond\psi$. We chose w_j to be a world maximal with respect to the property of satisfying ψ_j, hence Rw_iw_j. By the induction hypothesis, $\widehat{\mathfrak{M}}, w_j \Vdash \psi_j$. Hence $\widehat{\mathfrak{M}}, w_i \Vdash \Diamond\psi$. The converse implication is left to the reader. $\dashv$

Theorem 6.41 (Hemaspaandra's Theorem) *Every normal modal logic extending* $\mathbf{S}4.\mathbf{3}$ *has an NP-complete satisfiability problem.*

Proof. Lemma 6.40 established the polysize model property for Λ, so it remains to check that membership for Λ-frames can be decided in polynomial time. How can we show this? Recall Theorem 4.103:

> For every normal modal logic Λ extending **S4.3** there is a finite set N of finite **S4.3** frames with the following property: for any finite frame $\mathfrak{F}$, $\mathfrak{F} \Vdash \Lambda$ iff $\mathfrak{F}$ is an **S4.3** frame and there does not exist a bounded morphism from $\mathfrak{F}$ onto any frame in N.

This gives us a possible strategy: given any frame $\mathfrak{F}$, check whether it is an **S4.3** frame, and whether there is a surjective bounded morphism onto any frame in N. Now, as **S4.3** frames are first-order definable, by Lemma 6.36 the first part can be performed in polynomial time. But what about the second? First, note that because N is a *fixed* finite set, we need only ensure that the task of checking whether there is a bounded morphism from $\mathfrak{F}$ to a *fixed* frame $\mathfrak{G}$ can be performed in polynomial time. But the naive strategy of examining all the functions from $\mathfrak{F}$ to $\mathfrak{G}$ is completely unsuitable: the number of such functions is $|\mathfrak{G}|^{|\mathfrak{F}|}$, which is exponential in the size of $\mathfrak{F}$. However, applying Lemma 6.39, we see that the task can be simplified: we only need to check whether there is a set $\widehat{W}$ of worlds in $\mathfrak{F}$ such that $\mathfrak{F} \upharpoonright \widehat{W}$ is isomorphic to $\mathfrak{G}$ and $\widehat{W}$ contains a maximal world. Thus we need to check less than $|\mathfrak{F}|^{|\mathfrak{G}|}$ embeddings. But this number is *polynomial* in the size of $\mathfrak{F}$, for $\mathfrak{G}$ is fixed. By Lemma 6.35, NP-completeness follows. $\dashv$

The results of this section tell us something about the complexity of validity problems. The complement of NP is called co-NP. As a formula ϕ is not Λ-satisfiable iff $\neg\phi$ is Λ-valid, it follows that an NP-completeness result for Λ-satisfiability tells us that Λ-validity is co-NP complete (see Section C for further discussion). It is standardly conjectured that NP $\neq$ co-NP, thus the validity and satisfiability problems for these logics probably have different complexities.

Exercises for Section 6.6

6.6.1 Prove Lemma 6.36. That is, show that if F is a class of frames definable by a first-order sentence, then the problem of deciding whether $\mathfrak{F}$ belongs to F is decidable in time polynomial in the size of $\mathfrak{F}$.

6.6.2 Explain why the argument given in the proof of Lemma 6.35 may break down when we lift the restriction to finite similarity types. In particular, examine the situation when the similarity type contains modal operators of arbitrarily high arities.

6.6.3 Extend the proof of Theorem 6.38 to show that $\mathbf{K}_t\mathbf{Q}$ has the polysize model property, and is NP-complete.

6.6.4 Use a selection of points argument to show that **S5** has the polysize model property, and is NP-complete.

6.6.5 Show that if we restrict attention to a fixed finite set of proposition letters Φ, then the satisfiability problem for **S5** is decidable in linear time.
(Hint: if Φ is finite, the number of models we have to check to determine whether a given formula ϕ is satisfied in them, is independent of ϕ.)

6.7 PSPACE

PSPACE, the class of problems solvable by a deterministic Turing machine using only polynomial space, is the complexity class of most relevance to the basic modal language. As we will see, some important modal satisfiability problems belong to PSPACE, and many modal logics have PSPACE-hard satisfiability problems. This suggests that modal satisfiability problems are typically tougher than the satisfiability problem for propositional calculus, for it is standardly conjectured that PSPACE-hard problems are not solvable in NP.

The work of this section revolves around *trees*. We first show that **K** lacks the polysize model property by forcing the existence of binary-tree-based models using short formulas. We then take a closer look at **K**-satisfiability and show that it is in PSPACE. The proof also shows that every **K**-satisfiable formula is satisfiable on a tree-based model of polynomial *depth*. We then put all this work together to prove Ladner's theorem: every normal logic between **K** and **S4** has a PSPACE-hard satisfiability problem.

Forcing binary trees

The NP-completeness results of the previous section were proved using polysize model property arguments. So, before going any further, we will show that **K** does *not* have the polysize model property. We do so by showing that **K** can force the existence of binary trees. Many of the ideas introduced here will be reused in the proof of Ladner's theorem.

For any natural number m, we are going to devise a satisfiable formula $\phi^{\mathcal{B}}(m)$ with the following properties:

(i) the size of $\phi^{\mathcal{B}}(m)$ is polynomial (indeed, quadratic) in m, but
(ii) when $\phi^{\mathcal{B}}$ is satisfied in any model $\mathfrak{M}$ at a node w_0, then the submodel of $\mathfrak{M}$ generated by w_0 contains an isomorphic copy of the binary tree of depth m.

$$
\begin{array}{lllll}
\text{(i)}\ q_0 & & & & \\
\text{(ii)}\ \Box^{(m)}(q_i \to \bigwedge_{i \neq j} \neg q_j) \quad (0 \leq i \leq m) & & & & \\
\text{(iii)}\ B_0 \wedge \Box B_1 & \wedge \Box^2 B_2 & \wedge \Box^3 B_3 & \wedge \cdots \wedge \Box^{m-1} B_{m-1} & \\
\text{(iv)} \quad \Box S(p_1, \neg p_1) & \wedge \Box^2 S(p_1, \neg p_1) & \wedge \Box^3 S(p_1, \neg p_1) & \wedge \cdots \wedge \Box^{m-1} S(p_1, \neg p_1) & \\
 & \wedge \Box^2 S(p_2, \neg p_2) & \wedge \Box^3 S(p_2, \neg p_2) & \wedge \cdots \wedge \Box^{m-1} S(p_2, \neg p_2) & \\
 & & \wedge \Box^3 S(p_3, \neg p_3) & \wedge \cdots \wedge \Box^{m-1} S(p_3, \neg p_3) & \\
 & & & \vdots & \\
 & & & & \wedge \Box^{m-1} S(p_{m-1}, \neg p_{m-1})
\end{array}
$$

Fig. 6.5. The formula $\phi^{\mathcal{B}}(m)$.

As the binary branching tree of depth m contains 2^m nodes, the size of the smallest satisfying model of $\phi^{\mathcal{B}}(m)$ is exponential in $|\phi^{\mathcal{B}}(m)|$. Thus we will have shown that small formulas can force the existence of large models.

We will define these formulas by mimicking truth tables. For any natural number m, $\phi^{\mathcal{B}}(m)$ will be constructed out of the following variables: q_1, …, q_m, and p_1, …, p_m. The q_i play a supporting role. They will be used to mark the *level* (or *depth*) in the model; that is, they will mark the number of upward steps that need to be taken to reach the satisfying node. But any satisfying model for $\phi^{\mathcal{B}}(m)$ will give rise to a full truth table for $p_1, \ldots, p_m$: every possible combination of truth values for $p_1, \ldots, p_m$ will be realized at some node, and hence any model for $\phi^{\mathcal{B}}(m)$ must contain at least 2^m nodes.

That's the basic idea. To carry it out, we first define two macros: B_i, and $S(p_i, \neg p_i)$. For $i = 0, \ldots, m-1$, B_i is defined as follows:

$$B_i \quad := \quad q_i \to (\Diamond(q_{i+1} \wedge p_{i+1}) \wedge \Diamond(q_{i+1} \wedge \neg p_{i+1})) \,. \tag{6.12}$$

Given that we are going to use the q_is to mark the levels, the effect of B_i should be clear: it will force a *branching* to occur at level i, set the value of p_{i+1} to true at one successor at level $i+1$, and set p_{i+1} to false at another.

Our other macro is closely related. For $i = 0, \ldots, m-1$, $S(p_i, \neg p_i)$ is defined as follows:

$$S(p_i, \neg p_i) \quad := \quad (p_i \to \Box p_i) \wedge (\neg p_i \to \Box \neg p_i). \tag{6.13}$$

This formula *sends* the truth values assigned to p_i and its negation one level down. The idea is that once B_i has forced a branching in the model by creating a p_{i+1} and a $\neg p_{i+1}$ successor, $S(p_{i+1}, \neg p_{i+1})$ ensures that these newly set truth values are sent further down the tree; ultimately we want them to reach the leaves.

We are ready to define $\phi^{\mathcal{B}}(m)$. It is the conjunction of the formulas listed in Figure 6.5. Note that $\phi^{\mathcal{B}}(m)$ has the required effect. The first conjunct, q_0, ensures that any node that satisfies $\phi^{\mathcal{B}}(m)$ is marked as having level 0. The effect of (ii) is to ensure that no two distinct level marking atoms q_i and q_j can be true at the same node (at least, this will be the case all the way out to level m, which is all we care about). To see this, recall that $\Box^{(m)}\phi$ is shorthand for $\phi \wedge \Box\phi \wedge \Box^2\phi \wedge \cdots \wedge \Box^m\phi$. Thus our level markers are beginning to work as promised.

But the real work is carried out by (iii) and (iv). Because of the prefixed blocks of $\Box$ modalities, the B_i macros in (iii) force m successive levels of branching; and each such branching 'splits' the truth value of one of the p_is. Then, again because of the prefixed $\Box$ modalities, (iv) uses the $S(p_i, \neg p_i)$ macro to send each of these newly split truth values all the way down to the m-th level. In short, (iii) creates branching, and (iv) preserves it. It is worthwhile sitting down with a pencil and paper to check the details. If you do, it will become clear that $\phi^{\mathcal{B}}(m)$ is satisfiable, and that any satisfying model for $\phi^{\mathcal{B}}(m)$ must contain a submodel that is isomorphic to the binary branching tree of depth m. It follows that any model of $\phi^{\mathcal{B}}(m)$ must contain at least 2^m nodes, as we claimed.

In spite of its appearance, $\phi^{\mathcal{B}}(m)$ is indeed a *small* formula. To see this, consider what happens when we increment m by 1. The answer is: not much. For example (iii) simply gains an extra conjunct, becoming

$$\Box B_0 \wedge \Box^2 B_1 \wedge \Box^3 B_2 \wedge \cdots \wedge \Box^{m-1} B_{m-1} \wedge \Box^m B_m.$$

Similarly, each row in (iv) gains an extra conjunct (as does the next empty row) thus we gain a new column containing m formulas. The biggest change occurs in (ii). If you write (ii) out in full, you will see that it gains an extra row, and an extra column, and an extra atomic symbol in each embedded disjunct, and this means that the $|\phi^{\mathcal{B}}(m)|$ will increase is $O(m^2 \log m)$ (that is, slightly faster than quadratically). This is negligible compared with the explosion in the size of the smallest satisfying model: this doubles in size every time we increase m by one.

Theorem 6.42 **K** *lacks the polysize model property.*

That is, **K** lacks a property enjoyed by all the NP-complete logics examined in the previous section, and there is no obvious way of using NP guess-and-check algorithms to solve **K**-satisfiability. What sort of algorithms will work?

A PSPACE algorithm for K

We will now define a PSPACE-algorithm called *Witness* whose successful termination guarantees the **K**-satisfiability of the input. It may seem surprising that we can do this. After all, we have just seen that there are satisfiable formulas $\phi^{\mathcal{B}}(m)$ whose smallest satisfying model contains 2^m nodes. What happens if we give $\phi^{\mathcal{B}}(m)$ as input to *Witness*? Will it be forced to use an exponential amount of space to determine the satisfiability of $\phi^{\mathcal{B}}(m)$? The answer is: *no*. *Witness* will take an exponential amount of *time* to terminate on difficult input, but it uses *space* efficiently. As we will see, if a formula ϕ is satisfiable in some model, it is satisfiable in a tree-based model of polynomial depth. While some formulas require models with exponentially many nodes, we can always find a shallow satisfying model: the length of each branch is polynomial in $|\phi|$. *Witness* tests for the existence of shallow models, and does so one branch at a time. It does not need to keep track of the entire model, and hence can be made to run in PSPACE.

Witness is essentially an abstract tableaux system for **K**: it explores spaces of *Hintikka sets* (see Definition 6.24). Recall that Hintikka sets need not be satisfiable, and that we call satisfiable Hintikka sets *atoms*. *Witness* will take two finite sets of formulas H and Σ as input, and determine whether or not H is an atom over Σ. It does so by looking at the demands that H makes and recursively calculating whether all these demands can be met. The following definition makes the idea of a demand precise (compare Definition 4.62).

Definition 6.43 Suppose H is a Hintikka set over Σ, and $\Diamond\psi \in H$. Then the *demand* that $\Diamond\psi$ creates in H (notation: $Dem(H, \Diamond\psi)$) is

$$\{\psi\} \cup \{\theta \mid \Box\theta \in H\}.$$

We use $H_{\Diamond\psi}$ to denote the set of Hintikka sets over *Cl*$(Dem(H, \Diamond\psi))$ that contain $Dem(H, \Diamond\psi)$. (Recall that for any set of sentences Σ, *Cl*(Σ) denotes the closure of Σ; see Definition 6.23.) $\dashv$

Remark 6.44 Suppose that A is an atom over Σ, and that $\Diamond\psi \in A$. As A is satisfiable, so is $Dem(A, \Diamond\psi)$. From this it follows that there is at least one atom in $A_{\Diamond\psi}$ that contains $Dem(A, \Diamond\psi)$. For suppose $\mathfrak{M}, w \Vdash Dem(A, \Diamond\psi)$. Let Ψ be the set of all formulas satisfied in $\mathfrak{M}$ at w. Then $\Psi \cap$ *Cl*$(Dem(A, \Diamond\psi))$ is a an atom over *Cl*$(Dem(A, \Diamond\psi))$ that contains $Dem(A, \Diamond\psi)$.

Furthermore, as the reader can easily ascertain, for any formula ϕ, ϕ is satisfiable iff there is an atom A over *Cl*(ϕ) that contains ϕ. $\dashv$

Definition 6.45 Suppose H and Σ are finite sets of formulas such that H is a Hintikka set over Σ. Then $\mathcal{H} \subseteq$ *Pow*(Σ) is a *witness set generated by H on Σ* if $H \in \mathcal{H}$ and

(i) if $I \in \mathcal{H}$, then for each $\Diamond\psi \in I$, there is a $J \in I_{\Diamond\psi}$ such that $J \in \mathcal{H}$.
(ii) if $J \in \mathcal{H}$ and $J \neq H$ then for some $n > 0$ there are $I^0, \ldots, I^n \in \mathcal{H}$ such that $H = I^0$, $J = I^n$, and for each $0 \leq i < n$ there is some formula $\Diamond\psi \in I^i$ such that $I^{i+1} \in I^i_{\Diamond\psi}$.

The *degree* of a finite set of formulas Σ is simply the maximum of the degrees of the formulas contained in Σ; that is, $\deg(\Sigma) = \max\{\deg(\phi) \mid \phi \in \Sigma\}$. $\dashv$

For all choices of H and Σ, any witness set $\mathcal{H}$ generated by H on Σ must be finite, for $\mathcal{H} \subseteq Pow(\Sigma)$, which is a finite set. Further, observe that if $I, J \in \mathcal{H}$ and $J \in I_{\Diamond\psi}$ then the degree of J is strictly less than that of I. Moreover, observe that item (ii) of the previous definition is essentially a 'no junk' condition: if J belongs to $\mathcal{H}$, it is there because it is generated by some other elements of $\mathcal{H}$, and ultimately by H itself.

Lemma 6.46 *Suppose that H and Σ are finite sets of formulas such that H is a Hintikka set over Σ. Then H is an atom iff there is a witness set generated by H on Σ.*

Proof. For the left to right direction we proceed by induction on the degree of Σ. Let $\deg(\Sigma) = 0$, and suppose H is an atom. Trivially, $\mathcal{H} = \{H\}$ is a witness set generated by H. For the inductive step, suppose the required result holds for all pairs H' and Σ' such that H' is an atom of Σ' and $\deg(\Sigma') < n$. Let H be an atom of Σ such that $\deg(\Sigma) = n$. Then, as we noted in Remark 6.44, for all $\Diamond\psi \in H$ there exists at least one atom I^{ψ} in $H_{\Diamond\psi}$. As the degree of $Cl(Dem(H, \Diamond\psi)) < n$, for all $\Diamond\psi \in H$, the inductive hypothesis applies and every such atom I^{ψ} generates a witness set $\mathcal{I}^{\psi}$ on $Cl(Dem(H, \Diamond\psi))$. Define

$$\mathcal{H} = \{H\} \cup \bigcup_{\Diamond\psi \in H} \mathcal{I}^{\psi}.$$

Clearly $\mathcal{H}$ is a witness set generated by H on Σ.

For the right to left direction, we will show that if $\mathcal{H}$ is a witness set on Σ generated by H, then H can be satisfied in a model $(\mathfrak{F}, V)$ where $\mathfrak{F}$ is a finite tree of depth at most $\deg(H)$. This is stronger than the stated result, and later it will help us understand why **K**-satisfiability is solvable in PSPACE. Assume we have a countably infinite set of new entities $W = \{w_0, w_1, w_2, w_3, \ldots\}$ at our disposal. We will use (finitely many) elements of W to build a model for H, using a finitary version of the step-by-step method discussed in Section 4.6. This model will be a tree, thus showing once again that **K** has the tree model property.

Define $W_0 = \{w_0\}$, $R_0 = \varnothing$, $f_0(w_0) = H$. Suppose W_n, R_n and f_n have been defined. If for all $w \in W_n$ such that $\Diamond\psi \in f_n(w)$ there exists a $w' \in W_n$ such that (i) $\psi \in f_n(w')$ and (ii) $f_n(w') \in f_n(w)_{\Diamond\psi}$, then halt the step-by-step construction. Otherwise, if there is a $w \in W_n$ such that $\Diamond\psi \in f_n(w)$, while for no $w' \in W_n$ are these two conditions satisfied, then carry on to stage $n+1$ and define:

$$\begin{aligned} W_{n+1} &= W_n \cup \{w_{n+1}\}, \\ R_{n+1} &= R_n \cup \{(w, w_{n+1})\}, \\ f_{n+1} &= f_n \cup \{(w_{n+1}, I)\}, \end{aligned}$$

where $I \in \mathcal{H}$ is such that $I \in f_n(w)_{\Diamond\psi}$. Note that because $\mathcal{H}$ is a witness set it will always be possible to find such an I.

This step-by-step procedure halts after finitely many steps since each $I \in \mathcal{H}$ contains only finitely many formulas of the form $\Diamond\psi$ (thus ensuring that the tree we are constructing is finitely branching), and whenever $R_n ww'$, then $\deg(f_n(w')) < \deg(f_n(w))$ (thus ensuring that the tree is not only finite, but shallow: it has depth at most $\deg(H)$). Let m be the stage at which it halts, and define $\mathfrak{F}$ to be (W_m, R_m). To construct the desired model for H, it only remains to define a suitable valuation V, and we do this as follows: choose V to be any function from Σ to $\mathcal{P}(W_m)$ satisfying $w \in V(p)$ iff $p \in f_m(w)$, for all $p \in \Sigma$. Let $\mathfrak{M} = (\mathfrak{F}, V)$. Exercise 6.7.1 asks the reader to show that $\mathfrak{M}, w_0 \Vdash H$; an immediate consequence is that H is an atom. $\dashv$

Two remarks. The above proof shows that every atom is satisfiable in a shallow tree-based model — a fact which will prove to be important below. Second, we now have a syntactic criterion — namely the existence or non-existence of witness sets — for determining whether a Hintikka set is **K**-satisfiable. (In short, we have just proved a completeness result.) Moreover, the criterion is intuitively computable: witness sets are simple finite structures, thus it seems reasonable to expect that we can algorithmically test for their existence. And indeed we can.

We now define the *Witness* algorithm. This takes as input two finite sets of formulas H and Σ and returns the value **true** if and only if there is a witness set generated by H on Σ.

function $Witness(H, \Sigma)$ returns boolean
begin
  if H is a Hintikka set over Σ
    and for each subformula $\Diamond\psi \in H$ there is a set of formulas
      $I \in H_{\Diamond\psi}$ such that $Witness(I, \mathit{Cl}(Dem(H, \Diamond\psi)))$
  then return **true**
  else return **false**
end

Note that $Witness$ is an intuitively acceptable algorithm — and hence (by Church's thesis) implementable on a Turing machine. Checking that H is a Hintikka set over Σ involves ascertaining that Σ is closed, and that H satisfies the properties demanded of Hintikka sets; these tasks involve only simple syntactic checking. Moreover, both the '**and** for each subformula ... there is' clause and the recursive call to $Witness$ are clearly computable: the first involves search through a finite space, while the recursive call performs the same tasks on input of lower degree. Thus $Witness$ is indeed an algorithm. Moreover, it is *correct*: if H and Σ are finite sets of formulas, then $Witness(H, \Sigma)$ returns **true** iff H is Hintikka set over

Σ that generates a witness set in Σ. This follows by induction on the degree of Σ. The right to left direction is easy, while the left to right direction is similar to the proof of Lemma 6.46; see Exercise 6.7.2.

We are now ready for the main result.

Theorem 6.47 **K**-*satisfiability is in PSPACE.*

Proof. It follows from Lemma 6.46 and the correctness of $Witness$ that for any formula ϕ, ϕ is satisfiable iff there is an $H \subseteq Cl(\phi)$ such that $\phi \in H$ and $Witness(H, Cl(\phi))$ returns the value **true**. Thus, if we can show that $Witness$ can be given a PSPACE implementation, we will have the desired result. We will implement $Witness$ on a non-deterministic Turing machine. Given any formula ϕ, this machine will non-deterministically pick a Hintikka set H in $Cl(\phi)$ that contains ϕ, and run $Witness(H, Cl(\phi))$. It will be easy to show that this machine runs in non-deterministic PSPACE (that is, NPSPACE). But then it follows by an appeal to Savitch's Theorem ($\mathrm{PSPACE} = \mathrm{NPSPACE}$; see Section C) that the required PSPACE implementation exists.

So how do we implement $Witness$ on a non-deterministic Turing machine? The key points are the following:

(i) All sets of formulas used in the execution of the program are subsets of $Cl(\phi)$, and we can represent any such subset by using pointers to the connectives and proposition letters in ϕ's representation: a pointer to a propositional letter will mean that the letter belongs to the subset, and a pointer to a connective means that the subformula built using that connective belongs to it. Thus encoding a subset of $Cl(\phi)$ requires only space $\mathcal{O}(|\phi|)$ (that is, space of the order of the size of ϕ).

(ii) The '**and** for each subformula $\Diamond\psi \in H$' part can be handled by treating each subformula in turn. As any subformula can be represented using a pointer to ϕ's representation, we can cycle through all possible subformulas, using only polynomial space, by cycling through these pointers. Moreover, as we are using a non-deterministic Turing machine, the 'there is a set of formulas ...' clause can be implemented by making non-deterministic choices. Note that although $H_{\Diamond\psi}$ is a *set of* sets of formulas, to verify whether I belongs to it is a rather trivial task, given the definition of $H_{\Diamond\psi}$.

(iii) To enable the recursive calls to be made, we implement a stack on our Turing machine. To perform the recursion, we copy the formula ϕ onto the stack and point to propositional variables and connectives to indicate the subsets of interest.

So, suppose we run $Witness$ on input H and Σ. The crucial point that must be investigated is whether the recursive calls to $Witness$ cause a blow-up in space requirements. From items (i), (ii) and (iii) it is clear that at each level of recursion we use space $\mathcal{O}(|\phi|)$. How long does it take for the recursion to bottom out? Note that after $\deg(\phi)$ recursive calls, $\Sigma = \varnothing$. That is, the depth of recursion is bounded by $\deg(\phi)$ and hence by $|\phi|$. Thus, when we implement $Witness$ on a non-deterministic Turing machine the total amount of space required is $\mathcal{O}(|\phi|^2)$, hence the algorithm runs in NPSPACE. Thus, by Savitch's theorem, we conclude that **K**-satisfiability is in PSPACE. $\dashv$

The appeal to Savitch's theorem in the above proof can be avoided: $Witness$ can be implemented on a deterministic Turing machine. This involves replacing the non-deterministic choice used in item (iii) by brute force search through subsets of *Cl*(ϕ) that uses only polynomial space, and the reader is asked to do this in Exercise 6.7.4. But the above proof illustrates why Savitch's Theorem is so useful in practice: by freeing us to think in terms of non-deterministic computations, it reduces the required bookkeeping to a minimum.

Let us try and pin down the key intuition underlying Theorem 6.47. **K** lacks the polysize model property, but in spite of this the **K**-satisfiability problems can be determined in PSPACE. Why? The key lies in the proof of Lemma 6.46 which showed that every atom is satisfiable in a *shallow* finite tree-based model. Such models make it easy to visualize the explorations that $Witness$ makes as it tests the satisfiability of ϕ: it just works out what each branch of such a model must contain. While the size of the entire model may be exponential in $|\phi|$ it is not necessary to keep track of all this information. The locally relevant information is simply the information on each branch — and we know that the tree has depth at most $\deg(\phi) + 1$. In short, $Witness$ exploits the fact that only shallow tree-based models are needed to determine **K**-satisfiability.

PSPACE algorithms have been devised for a number of well-known logics including **T**, **K4** and **S4**, the temporal counterparts of **K**, **T**, **K4** and **S4**, and multimodal **K**, **T**, **K4**, **S4** and **S5**. While proofs of these results are essentially refinements of the proof Theorem 6.47, some are rather tricky. The reader who does Exercise 6.7.3, which asks for a PSPACE algorithm for **K4**, will find out why. In some cases alternative methods are preferable; see the Notes for pointers.

Ladner's theorem

We are ready to prove the major result of the section: every normal modal logic between **K** and **S4** is PSPACE-hard, and hence (assuming PSPACE $\neq$ NP) the satisfiability problems for all these logics are tougher than the satisfiability problem for propositional logic. We prove this by giving a polynomial time reduction of the validity problem for prenex quantified boolean formulas to all these modal satisfiability problems. The reduction boils down to forcing the existence of certain tree-based models, and we will be able to reuse much of our previous work.

Definition 6.48 The set of *quantified boolean formulas* is the smallest set X containing all formulas of propositional calculus such that if $\beta \in X$ and p is a proposition letter, then both $\forall p\, \beta$ and $\exists p\, \beta \in$ S. The quantifiers range over the truth values 1 (true) and 0 (false), and a quantified boolean formula without free variables is *valid* if and only if it evaluates to 1.

A quantified boolean formula is said to be in *prenex* form if it is of the form $\mathsf{Q}_1 p_1 \cdots \mathsf{Q}_m p_m\, \theta(p_1, \ldots, p_m)$; here Q is either $\forall$ or $\exists$, and $\theta(p_1, \ldots, p_m)$ is a formula of propositional logic. We will refer to such prenex formulas as QBFs. $\dashv$

The problem of deciding whether a QBF containing no free variables is valid is called the *QBF-validity problem*, and it is known to be PSPACE-complete.

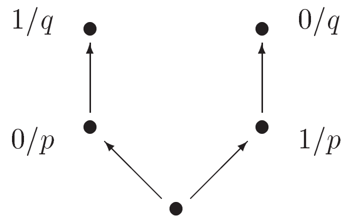
We are going to define a polynomial time translation f_L from QBFs to modal formulas, and prove that it has the following two properties:

(i) If β is a QBF-validity, then $f_L(\beta)$ is **S4**-satisfiable.
(ii) If $f_L(\beta)$ is **K**-satisfiable, then β is a QBF-validity.

These two properties — together with the known PSPACE-hardness of the QBF-validity problem — will lead directly to the desired theorem.

Let's think about what is involved in evaluating a QBF. We start by peeling off the outermost quantifier. If it is of the form $\exists p$ we choose one of the truth values 1 or 0 and substitute for the newly freed occurrences of p. On the other hand, if it is of the form $\forall p$ we must substitute both 1 and 0 for the newly freed occurrences of p. In this fashion, we work our way successively through the prefixed list of quantifiers until we reach the matrix, a formula of propositional logic.

Abstractly considered we are generating a tree. This tree consists of the root node, and then — working inwards along the quantifier string — each existential quantifier extends it by adding a single branch, and each universal quantifier extends it by adding two branches. Indeed, we are even generating an annotated tree: we can label each node with the substitution it records. For example, corresponding to the QBF $\forall p \exists q\, (p \leftrightarrow \neg q)$ we have the following annotated tree:



(i) q_0

(ii) $\Box^{(m)}(q_i \to \bigwedge_{i \neq j} \neg q_j) \quad (0 \leq i \leq m)$

(iiia) $\Box^{(m)}(q_i \to \Diamond q_{i+1}) \quad (0 \leq i < m)$

(iiib) $\bigwedge_{\{i \mid \mathsf{Q}_i = \forall\}} \Box^i B_i$

(iv)
$$\begin{aligned}\Box S(p_1, \neg p_1) \wedge \Box^2 S(p_1, \neg p_1) \wedge \Box^3 S(p_1, \neg p_1) \wedge \cdots \wedge \Box^{m-1} S(p_1, \neg p_1)\\ \wedge \Box^2 S(p_2, \neg p_2) \wedge \Box^3 S(p_2, \neg p_2) \wedge \cdots \wedge \Box^{m-1} S(p_2, \neg p_2)\\ \wedge \Box^3 S(p_3, \neg p_3) \wedge \cdots \wedge \Box^{m-1} S(p_3, \neg p_3)\\ \vdots\quad\\ \wedge \Box^{m-1} S(p_{m-1}, \neg p_{m-1})\end{aligned}$$

(v) $\Box^m(q_m \to \theta)$

Fig. 6.6. The formula $f_L(\beta)$.

The information in such annotated trees — we will call them *quantifier trees* — will play a crucial role. For a start, QBF-validity is witnessed by certain quantifier trees: β is a QBF-validity if and only if there is a quantifier tree for β such that the substitutions it records ensure that the matrix evaluates to 1. Moreover, quantifier trees give us a bridge between the QBF world and the modal world: $f_L(\beta)$ will be a modal formula that describes the structure of a quantifier tree evaluating β.

We define the translation f_L by modifying the way we forced the existence of binary trees in the proof of Theorem 6.42, and we will reuse the macros B_i and $S(p_1, \neg p_1)$ defined in (6.12) and (6.13), respectively.

Definition 6.49 Given any QBF $\beta = \mathsf{Q}_1 p_1 \cdots \mathsf{Q}_m p_m \, \theta(p_1, \ldots, p_m)$, choose new propositional variables $q_0, \ldots, q_m$. Then $f_L(\beta)$ is the conjunction of the formulas displayed in Figure 6.6. $\dashv$

The idea underlying f_L is this: for any QBF β, $f_L(\beta)$ describes the peel-of-quantifiers-and-substitute evaluation process for β. (That is, it describes how we generate a quantifier tree for β.) Moreover, it does so using ideas we have met already: note that (i), (ii) and (iv) are exactly the same formulas we used when forcing the existence of binary trees.

In fact, the major difference between these formulas and our earlier work lies in the word *binary*. Here we *don't* always want binary branching: we only want it when we encounter the quantifier $\forall$. Thus, instead of the earlier (iii) which forced branching all the way down to level m, we have the pair of formulas (iiia) and (iiib). (iiia) guarantees that if q_i is true and $i < m$ then there is a next level q_{i+1}; which simply amounts to saying that if $i < m$ then we have not yet peeled off all the quantifiers and a new level will be necessary. But it does *not* force binary branching. The task of forcing binary branching, when necessary, is left to (iiib). Note that this formula is simply a selection of conjuncts from our earlier (iii). There is only one other difference: (v) insists that after m quantifiers have been peeled off, the propositional matrix θ must be true.

Clearly, $f_L(\beta)$ is polysize in $|\beta|$, thus this translation causes no blowup in space requirements.

Theorem 6.50 (Ladner's Theorem) *If Λ is a normal modal logic such that* $\mathbf{K} \subseteq \Lambda \subseteq \mathbf{S4}$*, then Λ has a PSPACE-hard satisfiability problem. Moreover, Λ has a PSPACE-hard validity problem.*

Proof. Fix a modal logic Λ with $\mathbf{K} \subseteq \Lambda \subseteq \mathbf{S4}$. We are going to prove that f_L is a (polynomial time) reduction from the QBF-validity problem to the Λ-satisfiability problem. The crucial step in this proof is summarized in the following two statements:

$$\text{if } \beta \text{ is a QBF-validity, then } f_L(\beta) \text{ is satisfiable on a frame for } \mathbf{S4}, \tag{6.14}$$

and

$$\text{if } f_L(\beta) \text{ is satisfied in a } \mathbf{K}\text{-model then } \beta \text{ is a QBF-validity.} \tag{6.15}$$

From these two statements the desired result follows immediately. For suppose β is a QBF-validity. Then by (6.14) $f_L(\beta)$ is **S4**-satisfiable and hence Λ-satisfiable. Conversely, if $f_L(\beta)$ is Λ-satisfiable then it is also **K**-satisfiable, and by (6.15) β is a QBF-validity. Thus Λ-satisfiability is PSPACE-hard. That the Λ-validity problem is also PSPACE-hard follows immediately from the fact that PSPACE = co-PSPACE.

It remains to prove (6.14) and (6.15). For (6.14), assume that β is a QBF-validity. Generate a quantifier tree witnessing the validity of β; if β is valid, such a tree must exist. This tree gives rise to an **S4**-model for $f_L(\beta)$ as follows. First, take the transitive and reflexive closure of the 'daughter-of' relation of the tree; this gives us the **S4**-frame we require. Then make the variable q_i true precisely at the nodes of level i; p_i is to be made true at a node of level $j \geq i$ iff the substitution connected to that node, or its predecessor at level i returns the value 1 for p_i. (For nodes at level $j < i$ it does not matter what truth value we choose for p_i.) It is straightforward to check that the formula $f_L(\beta)$ is true in this model at the root of the tree; see Exercise 6.7.5.

For (6.15), suppose that β is a QBF of quantifier depth m, and that $f_L(\beta)$ is **K**-satisfiable. Note that $\deg(f_L(\beta)) = m$, hence from the proof of Lemma 6.46 we know that $f_L(\beta)$ holds at the root r of a tree-based model $\mathfrak{M} = (T, R, V)$ of depth at most m. Using clauses (iiia) and (iiib) of the definition of $f_L(\beta)$, it is easily verified that we may cut off branches from this tree such that in the resulting tree, a node at level $i < m$ has either one or two successors. This number is one iff $Q_{i+1} = \exists$. And if $Q_{i+1} = \forall$, then one of the successors satisfies p_{i+1} and the other one, $\neg p_{i+1}$. But then this reduced tree model is a quantifier tree witnessing the validity of β. $\dashv$

Among other things, Ladner's theorem tells us that **K**, **T**, **K4** and **S4** have PSPACE-hard satisfiability problems. It follows that the temporal counterparts of **K**, **T**, **K4** and **S4**, and multi-modal **K**, **T**, **K4**, and **S4**, are PSPACE-hard too, for they contain the unimodal satisfiability problems as a special case. Hence, as PSPACE algorithms are known for these logics, they all have PSPACE-complete satisfiability problems. As PSPACE = co-PSPACE, these logics have PSPACE-complete validity problems too.

Exercises for Section 6.7

6.7.1 Show that in the model $\mathfrak{M}$ constructed in the proof of Lemma 6.46, $\mathfrak{M}, w_0 \Vdash H$.

6.7.2 We claimed that *Witness* is a correct algorithm. That is, if H and Σ are finite sets of formulas, then $Witness(H, \Sigma)$ returns **true** iff H is Hintikka set over Σ that generates a witness set in Σ. Prove this.

6.7.3 Adapt the *Witness* algorithm so that it decides **K4** satisfiability correctly. (Hint: since you can't consider smaller and smaller Hintikka sets (why not?) make use of lists of Hintikka sets, rather than the single Hintikka sets used in the proof for **K**, and show that the length of such lists can always be kept polynomial.)

6.7.4 Show how to avoid the use of Savitch's Theorem in the proof of Theorem 6.47. That is, show that the $Witness$ function can be implemented on a *deterministic* Turing machine. (Hint: implement the '**and** for each subformula ... there is' clause by cycling through all possible subsets of *Cl*(ϕ). This cycling process has a simple implementation using only space $\mathcal{O}(|\phi|)$: generate all binary strings of length $|\phi|$, and decide of each whether or not it encodes a subset of *Cl*(ϕ).)

6.7.5 Supply the missing details in the proof of Ladner's Theorem.

6.7.6 Show that the satisfiability problem for bimodal **S5** is PSPACE-hard.

6.7.7 In this exercise we examine the effects of bounding the number of proposition letters and of restricting the degree of formulas.

(a) Show that for any fixed k, the satisfiability problem for **K** with respect to a language consisting of all formulas whose degree is at most k, is NP-complete.
(b) Show that, in contrast, the satisfiability problem for **S4** remains PSPACE-complete for languages consisting of all formulas of degree at most k ($k \geq 2$).
(c) Now suppose that Φ, the set of proposition letters, is finite. Show that for any fixed k, the satisfiability problems for **K** and **S4** with respect to a language consisting of all formulas whose degree is at most k, is decidable in linear time.

6.8 EXPTIME

EXPTIME, the class of problems deterministically solvable in exponential time, is an important complexity class for many modal languages. In particular, when a modal language has operators $[a]$ and $[a^*]$ which explore a relation R_a and its reflexive transitive closure $(R_a)^*$, its satisfiability problem is likely to be EXPTIME-hard, which means that the worst cases are computationally intractable. As such operator pairs are important in many applications, we need to understand the complexity theoretic issues they give rise to. In this section we examine the satisfiability problem for **PDL**; our discussion illustrates some key themes and introduces some useful techniques.

Forcing exponentially deep models

By Corollary 6.14 we know that **PDL** has a decidable satisfiability problem — but just how difficult is it? Clearly it is PSPACE-hard, for each basic modality $[a]$ is a **K** operator, and we saw in the previous section (Theorem 6.50) that **K** has a PSPACE-hard satisfiability problem. But can we prove a matching PSPACE upper bound?

We used a tableaux-like algorithm called $Witness$ to show that **K**-satisfiability was solvable in PSPACE. $Witness$ traded on the following insight: while a **K**-consistent formula ϕ may require a satisfying model of size $2^{|\phi|}$, it is always possible to build a satisfying tree model of this size in which each branch has less than $|\phi|$ nodes. $Witness$ tests for **K**-satisfiability by building such trees one branch at a time; as each branch is polynomial in the size of the input, $Witness$ runs in PSPACE. However, as we will now show, even small fragments of PDL are strong enough to force the existence of exponentially *deep* models.

Proposition 6.51 *For every natural number n there is a satisfiable* PDL *formula κ_n of size $O(n^2)$ such that every model which satisfies κ_n contains an R_a-path containing 2^n distinct nodes. Moreover, κ_n contains occurrences of only two modalities $[a]$ and $[a^*]$, where a is an atomic program.*

Proof. We will show how to count using this PDL-fragment. Given a natural number n, we select n distinct proposition letters $q_1, \ldots, q_n$. Using 1 for true, and 0 for false, the list of truth values $[V(q_n, w), \ldots, V(q_i, w), \ldots, V(q_1, w)]$ is the n-bit binary encoding of a natural number. We take $V(q_1, w)$ to be the least significant digit, and $V(q_n, w)$ to be the most significant.

We now construct a formula κ_n which, when satisfied at some state w_0, forces the (n-bit representation of) zero to hold at w_0, and forces the existence of a path of distinct successors of w_0 which correctly count from 0 to 2^{n-1} in binary. For example, if $n = 2$, the model will contain a path of length 4 from w_0 to w_3, and as we move along this path we will successively encounter the following truth value lists: $[0,0],[0,1],[1,0],[1,1]$.

To do the encoding, we need to know what happens when we add 1 to a binary number m. First suppose that the least significant bit of m is 0; for example, suppose that m is 010100. When we add 1 we obtain 010101; that is, we flip the least significant digit to 1 and leave everything else unchanged. We can force this kind of incrementation in PDL as follows:

$$\mathit{INC}_0 \;:=\; \neg q_1 \to \left([a]q_1 \wedge \bigwedge_{j>1}((q_j \to [a]q_j) \wedge (\neg q_j \to [a]\neg q_j)\right).$$

This guarantees that the value of q_1 changes to 1 at any successor state, while the truth values of all the other q_js remain unchanged.

Now suppose that the least significant digit of m is 1. For example, suppose that m is 01011. When we add 1 we obtain 01100. We can describe this incrementation as follows. First, we locate the longest unbroken block of 1s containing the least significant digit and flip all these 1s to 0s. Second, we flip the following digit from 0 to 1 (we have to 'carry one'). Finally, we leave all remaining digits unchanged. The following formula forces this kind of incrementation when the longest unbroken block of 1s containing the least significant digit has length i, where $0 < i < n$:

$$\begin{aligned}&\mathit{INC}_1(i) :=\\ &\quad\left(\neg q_{i+1} \wedge \bigwedge_{j=1}^{i} q_j\right) \to\\ &\quad\left([a](q_{i+1} \wedge \bigwedge_{j=1}^{i} \neg q_j \wedge \bigwedge_{k>i+1}((q_k \to [a]q_k) \wedge (\neg q_k \to [a]\neg q_k))\right).\end{aligned}$$

We can now define the required formula κ_n:

$$(\neg q_n \wedge \cdots \wedge \neg q_1) \wedge [a^*]\langle a\rangle\top \wedge [a^*]\left(\mathit{INC}_0 \wedge \bigwedge_{i=1}^{n-1} \mathit{INC}_1(i)\right).$$

The first conjunct of κ_n initializes the counting at 0, the second guarantees that there will always be successor states, while the third guarantees that incrementation is carried out correctly. Clearly κ_n is of size $O(n^2)$ and uses only the allowed modalities. $\dashv$

Proposition 6.51 is suggestive. It does not *prove* that no PSPACE algorithm is possible, but it does tend to confirm our suspicions that **PDL**-satisfiability is computationally difficult. And indeed it is. The remainder of the chapter is devoted to proving the following result: **PDL**-satisfiability problem is EXPTIME-complete.

The proof methods we use are important in their own right and well worth mastering: we will prove EXPTIME-hardness by reduction from the two person corridor tiling game, and demonstrate the existence of an EXPTIME algorithm using elimination of Hintikka sets.

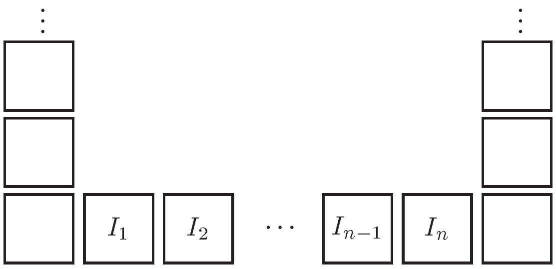
EXPTIME-hardness via tiling

In Section 6.5 we used tiling problems to prove two undecidability results. We remarked that tiling problems were also useful for proving complexity results, and in this section we give an example. We will describe the *two person corridor tiling game* and use it to prove the EXPTIME-hardness of **PDL**-satisfiability; we make use of notation and ideas introduced in our discussion of undecidability.

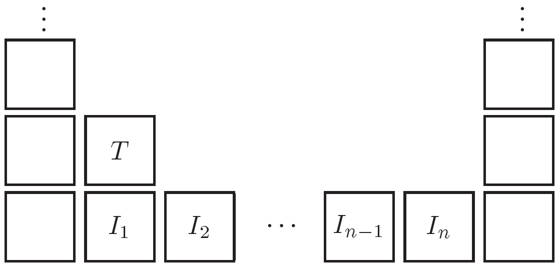
As with our earlier tiling games, the two person corridor tiling game involves placing tiles on a grid so that colors match, but there are some extra ingredients. There are two players, and we assume that there is a third person present — the referee — who starts the game correctly and keeps it flowing smoothly. The referee will give the players a finite set $\{T_1, \ldots, T_s\}$ of tile types; the players will use tiles of these types to attempt to tile a grid so that colors match. In addition, the referee will set aside two special tile types: T_0 and T_{s+1}. T_0 is there solely to mark the boundaries of the corridor (we think of the boundaries as having some distinctive color, say white), while T_{s+1} is a special winning tile, whose role will be described later.

At the start of play, the referee places n initial tiles $I_1, \ldots, I_n$ in a row. To the left of I_1 and to the right of I_n he places copies of the white tile T_0. That is, the following sequence of tiles is the initial position:

This is the first row of the corridor. The white tiles in column 0 and column $n+1$ mark the boundaries of the corridor. Columns 1 through n are the corridor proper. Actually, we may as well stipulate that the referee immediately fills in columns 0 and $n+1$ with the special boundary-marking white tile. That is, the players are going to be playing into the grid inside the following n-column corridor:



Now the players are ready to start. There are two players, Eloise and Abelard. The players take turn placing tiles in the corridor, and it's always Eloise who moves first. The rules for tile placement are strict: the corridor has to be filled in from the bottom, from left to right. For example, after Eloise has placed her first tile the corridor will look like this:



When Abelard replies, he must place his tile immediately to the right of tile T. When the players have completed tiling a row, they start tiling the next one, starting at column 1. In short, the players have no choice about *where* to place a tile, only about which type of tile they will place there. The player's choice of tiles is subject to the usual color-matching rules of tiling, and any tile placed in column 0 or column n has to match the white of the corridor tile. (For example, in the previous diagram, it must be the case that *left*$(T) =$ *white*.)

When do the players win or lose? As follows. If after finitely many rounds a tiling is constructed in which the special winning tile T_{s+1} is placed in column 1, Eloise wins. Otherwise (that is, if one of the players can't make a legal move and T_{s+1} is not in column 1, or if the game goes on infinitely long) Abelard wins.

Now for the EXPTIME-complete problem: *given a game, does Eloise have a winning strategy in that game*? That is, can she win the game no matter what Abelard does? It is useful to think of winning strategies in terms of *game trees*. For any game, a game tree for that game records all possible responses Abelard can make to Eloise's moves. (Note that we don't insist that game trees encode all of Eloise's options; but it is *vital* that game trees record all of Abelard's options.) Note that Abelard has only finitely many possible responses, for there are only finitely many tile types. Clearly, if Eloise has a winning strategy in a game, then there is a game tree that describes that strategy: such a tree spells out exactly what she has to do, and takes all Abelard's possible responses into account.

Now that we know about game trees, let's think about winning strategies for Eloise. In fact, we can recursively characterize this concept. We first define the notion of a *winning position for Eloise* in a game tree:

(i) Whenever the winning tile T_{s+1} is placed in column 1, that position is a winning position for Eloise.
(ii) In case Eloise is to move in position x, then x is a winning position for Eloise if there exists a move to a winning position for Eloise.

(iii) In case Abelard is to move in position x, then x is a winning position for Eloise if Abelard can make a move and all his moves lead to a winning position for Eloise.

We now say that Eloise has a *winning strategy* iff there is a game tree such that the root of the game tree is a winning position for her. The problem of determining whether Eloise has a winning strategy is called the *two person corridor tiling problem*, which is known to be EXPTIME-complete (see the Notes for references).

Theorem 6.52 *The satisfiability problem for* **PDL** *is EXPTIME-hard.*

Proof. We show this by reducing the two person corridor tiling problem to the **PDL** satisfiability problem. We will view a game tree as a rooted regular PDL model with one atomic transition R_m which codes one move of the game. Given an instance $\mathcal{T} = (n, \{T_0, \ldots, T_{s+1}\})$ of the two person corridor tiling game (here n is the width of the corridor, and the T_i are the tile types), we will show how to create a formula $\phi_{\mathcal{T}}$ such that

(i) If Eloise has a winning strategy, $\phi_{\mathcal{T}}$ is satisfiable at the root of some game tree for $\mathcal{T}$ (viewed as a regular PDL model).
(ii) If $\phi_{\mathcal{T}}$ is satisfiable, then Eloise has a winning strategy in the game T; in fact, she will be able to read off her winning strategy by following a path through the satisfying model (starting at the point that satisfies $\phi_{\mathcal{T}}$).
(iii) The formula $\phi_{\mathcal{T}}$ can be computed in time polynomial in n and s.

The formula $\phi_{\mathcal{T}}$ contains two kinds of information: it fully describes the structure of the game tree, and states necessary and sufficient conditions for Eloise to win. The first part boils down to using PDL to describe the initial configuration, that players move alternately, that colors match, and so on; this is a little tedious, but straightforward. Stating necessary and sufficient conditions for Eloise to win involves finding PDL formulas that capture the recursive characterization of winning strategies, and prevent the game from running for infinitely many moves; this is the interesting part of the proof.

We use the following proposition letters to construct $\phi_{\mathcal{T}}$:

(i) t_0, t_1, …, t_s, and t_{s+1}. These will be used to represent the tiles. We will often write t_0 as *white*.
(ii) *eloise*. This will be used to indicate that Eloise has the next move. Its negation will indicate that Abelard has the next move.
(iii) $pos_1, \ldots, pos_n$. We use pos_i to indicate that in the current round, a tile is to be placed in column i.
(iv) $col_i(t)$, for all $0 \leq i \leq n+1$ and all $t \in \{t_0, t_1, \ldots, t_s, t_{s+1}\}$. These will be used to indicate that the tile previously placed in column i is of type t.

(v) *win*. This means that the current position is a winning position for Eloise.

In addition, we make use of the modalities $[m]$ and $\langle m\rangle$ ('after every possible move' and 'after some possible move' respectively) and $[m^*]$, which can be read as 'after every possible sequence of moves'.

So let's describe the structure of the game tree. The following formula records the situation at the start of play:

$$\mathit{eloise} \wedge \mathit{pos}_1 \wedge \mathit{col}_0(\mathit{white}) \wedge \mathit{col}_1(t_{I_1}) \wedge \cdots \wedge \mathit{col}_n(t_{I_n}) \wedge \mathit{col}_{n+1}(\mathit{white}).$$

The first conjunct says that Eloise has to make the first move, while the second says that she has to place her tile in column 1. The remaining conjuncts simply say that the tiles previously placed in all columns are those of the initial configuration. (Of course, these were not placed by the players but by the referee.) That is, they say that columns 1 through n contain the initial tiles $I_1, \ldots, I_n$, and that there is a white corridor tile on each side.

We now write down a series of formulas which regulate the way that further play takes place. (Note that all these conditions are preceded by the $[m^*]$ modality, thus ensuring that they continue to hold after any finite sequence of moves.) We start by giving the desired meaning to pos_i and $col_i(t)$.

- Tiles always have to be placed in one of columns 1 through n:

  $$[m^*](\mathit{pos}_1 \vee \cdots \vee \mathit{pos}_n),$$

  and indeed, in *exactly* one of these columns:

  $$[m^*](\mathit{pos}_i \rightarrow \neg\mathit{pos}_j) \qquad (1 \leq i \neq j \leq n).$$

- In every column i, at least one tile type was previously placed:

  $$[m^*](\mathit{col}_i(t_0) \vee \cdots \vee \mathit{col}_i(t_{s+1})) \qquad (0 \leq i \leq n+1).$$

- In every column i, at most one tile type was previously placed:

  $$[m^*](\mathit{col}_i(t_u) \rightarrow \neg\mathit{col}_i(t_v)) \qquad (0 \leq i \leq n+1 \text{ and } 0 \leq u \neq v \leq s+1).$$

- Moreover, the referee has already placed white tiles in columns 0 and $n+1$:

  $$[m^*](\mathit{col}_0(\mathit{white}) \wedge \mathit{col}_{n+1}(\mathit{white})).$$

- In the course of play, tiles are placed left-to-right (flipping back to column 1 when a row has been completed):

  $$[m^*]((\mathit{pos}_1 \rightarrow [m]\mathit{pos}_2) \wedge (\mathit{pos}_2 \rightarrow [m]\mathit{pos}_3) \wedge \cdots \wedge (\mathit{pos}_n \rightarrow [m]\mathit{pos}_1)).$$

- In columns where no tile is placed, nothing changes when a move is made:

  $$[m^*](\neg\mathit{pos}_i \rightarrow ((\mathit{col}_i(t_u) \rightarrow [m]\mathit{col}_i(t_u)) \wedge (\neg\mathit{col}_i(t_u) \rightarrow [m]\neg\mathit{col}_i(t_u)).$$

  (Here $0 \leq i \leq n+1$ and $0 \leq u \leq s+1$.)

With these preliminaries behind us, we can now describe the structure of the game tree.

- First of all, players alternate:

$$[m^*]((\mathit{eloise} \to [m]\neg \mathit{eloise}) \land (\neg \mathit{eloise} \to [m]\mathit{eloise})).$$

- Next, both players make legal moves; that is, they only place tiles which correctly match adjacent tiles. It will be helpful to define the following ternary relation of 'compatibility' between propositional variables:

$$C(t', t, t'') \text{ iff } \mathit{right}(T') = \mathit{left}(T) \text{ and } \mathit{down}(T) = \mathit{up}(T''),$$

  where T, T' and T are the tiles that correspond to the propositional variables t, t' and t'' respectively. That is, $C(t', t, t'')$ holds iff the tile T can be placed to the right of tile T' and above tile T''. With the aid of this relation we can formulate the first constraint on tile placement as follows:

$$[m^*]\left(\mathit{pos}_i \land \mathit{col}_{i-1}(t') \land \mathit{col}_i(t'') \to [m]\bigvee\{\mathit{col}_i(t) \mid C(t', t, t'')\}\right).$$

  (Here $0 \leq i \leq n$, and, by convention, $\bigvee \varnothing = \bot$.)
- However this constraint is not quite enough; it only ensures matching to the left and downwards. We also need to ensure that tiles placed in column n match the white corridor tile to their right, and we can do this as follows:

$$[m^*]\left(\mathit{pos}_n \to [m]\bigvee\{\mathit{col}_n(t) \mid \mathit{right}(T) = \mathit{white}\}\right).$$

  (Here t is the proposition letter corresponding to tile T.)
- Next, we need to ensure that all of Abelard's possible responses are encoded in the model:

$$[m^*]\left(\neg \mathit{eloise} \land \mathit{pos}_i \land \mathit{col}_i(t'') \land \mathit{col}_{i-1}(t') \to \bigwedge\{\langle m\rangle \mathit{col}_i(t) \mid C(t', t, t'')\}\right).$$

  (Here $1 \leq i < n$, and, by convention, $\bigwedge \varnothing = \top$.)

That completes our description of the game tree. So let's turn to our other task: ensuring that Eloise indeed has a winning strategy. We will do this with the help of our recursive characterization of winning strategies, thus the first step is easy; we simply state that the initial position is a winning position for Eloise:

$$\mathit{win}.$$

Next, we spell out the recursive conditions:

$$[m^*]\left(\mathit{win} \to (\mathit{col}_1(t_{s+1}) \lor (\neg \mathit{eloise} \land \langle m\rangle\top \land [m]\mathit{win}) \lor (\mathit{eloise} \land \langle m\rangle \mathit{win}))\right).$$

We're almost there — but we don't have quite enough. If a game does not terminate, Abelard wins, so we need to rule out this possibility. Now, any infinite branch must involve repetition of rows. Indeed, if $N = n^{s+2}$, then if a game runs N moves, repetition must have occurred.

Repetitions do not help Eloise: if she can win, she can do so in fewer than N moves. So we are simply going to insist that games run fewer than N moves — and we can do this with the help of the PDL counter defined in the proof of Proposition 6.51. To use the notation of that proof, we make use of propositional variables $q_1, \ldots, q_n$, all initially set to zero, and increment the counter by 1 at every move. If the counter reaches N (that is, if all these propositional variables are true in some successor state) then the game has gone on too long and Abelard wins. The following formula encodes this observation:

$$[m^*]((\mathit{counter} = N) \to [m]\neg \mathit{win}).$$

Let $\phi^{\mathcal{T}}$ be the conjunction of all these formulas. We must now verify the three claims made about $\phi^{\mathcal{T}}$ at the start of the proof.

First we need to show that if Eloise has a winning strategy, then there is a game tree such that $\phi_{\mathcal{T}}$ is satisfiable at the root of the game tree viewed as a PDL model. If Eloise has a winning strategy, then she can win in at most N moves. If $\mathfrak{M}$ is the PDL model corresponding to this at-most-N move strategy, then it is straightforward to check that $\phi^{\mathcal{T}}$ is satisfied at the root of $\mathfrak{M}$.

The second claim is more interesting: we need to show that if $\mathfrak{M}, w \Vdash \phi_{\mathcal{T}}$, then Eloise has a winning strategy in the game T — and that her winning strategy is encoded in $\mathfrak{M}$. So suppose there is such a model. Imagine Eloise facing Abelard across the playing board and consulting this model to choose her moves. As $\phi^{\mathcal{T}}$ is satisfied at w, *win* is satisfied at w (remember that the initial position is marked as winning), hence $eloise \wedge \langle m \rangle win$, the third disjunct of our recursive characterization of winning strategy, is true at w too. Eloise simply needs to pick a successor state in the model marked as winning to see which tile to place. In short, she plays the move described in the model, and continues doing so in subsequent rounds.

Though this guarantees that Eloise can keep moving to winning positions, can she actually win the game after finitely many moves? Yes! In fact she can win in at most N moves. For suppose the N-th move has just been played (that is, $\mathit{counter} = N$ has just become true). As Eloise has always been moving to winning positions, the N-th position is also winning, which means that one of the following formulas is satisfied there:

- $\mathit{col}_1(t_{s+1})$, or
- $\neg \mathit{eloise} \wedge \langle m \rangle \top \wedge [m]\mathit{win}$, or
- $\mathit{eloise} \wedge \langle m \rangle \mathit{win}$.

As the counter has reached N, $[m]\neg \mathit{win}$ is satisfied too. This means there aren't any more winning positions, and so the second and third disjuncts are false. Hence

$col_1(t_{s+1})$ is satisfied: the winning tile was placed in the first column in the previous round. Thus Eloise has already won.

It remains to check that $\phi^{\mathcal{T}}$ is polynomial in n and s. The only point that requires comment is that we can encode N. Encoding any natural number $m \geq 2$ in binary requires at most $lg(m) + 1$ bits (lg denotes the logarithm to base 2). So encoding N takes at most $lg(n^{s+2}) = (s+2)lg(n) \leq (s+2)n$ bits, which is polynomial in n and s. Thus we have reduced the two person corridor tiling problem to the satisfiability problem for **PDL**, hence the latter is EXPTIME-hard. $\dashv$

As the previous proof makes clear, the EXPTIME-hardness of **PDL** largely stems from the fact that it contains a pair of modalities, one for working with a relation R_m and the other for reflexive transitive closure $(R_m)^*$; this is what enabled us to force exponentially deep models, and to code the corridor tiling problem. Now, this is not the entire story, for there *are* logics containing such modality pairs whose satisfiability problem is in PSPACE: one example is the modal logic of the frame $(\mathbb{N}, S, <)$, in a language with two diamonds $\langle s \rangle$ and $\langle < \rangle$. Here $\mathbb{N}$ is the set of natural numbers, S is the successor-of relation, and $<$ is the usual ordering of $\mathbb{N}$. An even more expressive language — one involving the until operator — has a PSPACE-complete satisfiability problem over $(\mathbb{N}, S, <)$; see the Notes for references.

Despite this, the following is a reliable rule of thumb: when working with a modal language containing a pair of modalities for working with a relation and its transitive closure, suspect EXPTIME-hardness. Don't begin your investigations by looking for a PSPACE-algorithm, unless you are working with a class of frames that allows little or no branching. And as this section has demonstrated, an elegant way of proving EXPTIME-hardness is via the two person corridor tiling game.

Elimination of Hintikka sets

By Theorem 6.52 there are instances of the **PDL**-satisfiability problem which will require exponentially many steps to solve. As yet we have no matching upper bound. In fact, so far the best solution to **PDL**-satisfiability we have is the following *nondeterministic* algorithm: given a formula ϕ, let Σ be the set of all ϕ's subformulas, form the collection of all Hintikka sets in Σ, nondeterministically choose a model of size at most $2^{c|\phi|}$, and check ϕ on this model. By the decidability result for **PDL** (Corollary 6.14), if ϕ is satisfiable, it is satisfiable in a model of at most this size, hence **PDL**-satisfiability is solvable in NEXPTIME.

As we will now show, the EXPTIME-hardness result of the previous section can be matched by an EXPTIME algorithm. Like the PSPACE $Witness$ algorithm developed in the previous section, the EXPTIME algorithm for **PDL** is based around the idea of *Hintikka sets*. Here's how we define this notion for **PDL**.

Definition 6.53 (Hintikka set for PDL) Let Σ be a set of PDL formulas and $\neg FL(\Sigma)$ the closure under single negations of its Fisher-Ladner closure (see Definition 4.79). A *Hintikka set* over Σ is any maximal subset of $\neg FL(\Sigma)$ that satisfies the following conditions:

(i) If $\neg\phi \in \neg FL(\Sigma)$, then $\neg\phi \in H$ iff $\phi \notin H$.
(ii) If $\phi \wedge \psi \in \neg FL(\Sigma)$, then $\phi \wedge \psi \in H$ iff $\phi \in H$ and $\psi \in H$.
(iii) If $\langle \pi_1; \pi_2 \rangle\phi \in \neg FL(\Sigma)$, then $\langle \pi_1; \pi_2 \rangle\phi \in H$ iff $\langle \pi_1 \rangle\langle \pi_2 \rangle\phi \in H$.
(iv) If $\langle \pi_1 \cup \pi_2 \rangle\phi \in \neg FL(\Sigma)$, then $\langle \pi_1 \cup \pi_2 \rangle\phi \in H$ iff $\langle \pi_1 \rangle\phi$ or $\langle \pi_2 \rangle\phi \in H$.
(v) If $\langle \pi^* \rangle\phi \in \neg FL(\Sigma)$, then $\langle \pi^* \rangle\phi \in H$ iff $\phi \in H$ or $\langle \pi \rangle\langle \pi^* \rangle\phi \in H$.

We denote the set of all Hintikka sets over Σ by $Hin(\Sigma)$. $\dashv$

The first clause of Definition 6.53 ensures the maximality of Hintikka sets: if $H \in Hin(\Sigma)$ then there is no $H' \in Hin(\Sigma)$ such that $H \subset H'$. So, when the effect of clause (ii) is taken into account, we see that Hintikka sets are maximal subsets of $\neg FL(\Sigma)$ that contain no blatant propositional inconsistencies. Hintikka sets for PDL are a generalization of something we met in Chapter 4, namely *atoms* (see Definition 4.80). Clearly $At(\Sigma) \subseteq Hin(\Sigma)$; indeed, $At(\Sigma)$ contains precisely the **PDL**-consistent Hintikka sets.

We use Hintikka sets as follows. We define a model $\mathfrak{M}^0$ that is built out of $Hin(\Sigma)$. We then iteratively *eliminate Hintikka sets* from this model, thus forming a sequence of ever smaller models. This process is deterministic, and terminates after at most exponentially many steps yielding a model $\mathfrak{M}$. We will then show that a PDL formula ϕ is satisfiable iff it is satisfiable in $\mathfrak{M}$.

Elimination of Hintikka sets:

Base case. Let Σ be a finite set of PDL formulas, and let Π be the set of programs that occur in Σ. Define W^0 to be $Hin(\Sigma)$. For all basic programs a, and all $H, H' \in W^0$, define a binary relation Q^0_a by HQ^0_aH' iff for every $\phi \in H'$, if $\langle a \rangle\phi \in \neg FL(\Sigma)$ then $\langle a \rangle\phi \in H$. For all other programs $\pi \in \Pi$, define Q^0_π to be the usual inductively defined PDL relations, and let $\mathfrak{F}^0$ be $(W^0, Q^0_\pi)_{\pi\in\Pi}$. Define V^0 by $V^0(p) = \{H \in W^0 \mid p \in H\}$, for all propositional variables p. Finally, let $\mathfrak{M}^0$ be $(\mathfrak{F}^0, V^0)$.

Inductive step. Suppose that $n \geq 0$ and that $\mathfrak{F}^n = (W^n, Q^n_\pi)_{\pi\in\Pi}$ and $\mathfrak{M}^n = (\mathfrak{F}^n, V^n)$ are defined. Say that $H \in W^n$ is *demand-satisfied* iff for all $\pi \in \Pi$, and all formulas ψ, if $\langle \pi \rangle\psi \in H$ then there is an $H' \in W^n$ such that $HQ^n_\pi H'$ and $\psi \in H'$. Then define:

(i) $W^{n+1} = \{H \in W^n \mid H \text{ is demand-satisfied}\}$.
(ii) Q^{n+1}_π is $Q^n_\pi \cap (W^{n+1} \times W^{n+1})$, and $\mathfrak{F}^{n+1}$ is $(W^{n+1}, Q^{n+1}_\pi)_{\pi\in\Pi}$.
(iii) V^{n+1} is $V^n \upharpoonright W^{n+1}$, and $\mathfrak{M}^{n+1}$ is $(\mathfrak{F}^{n+1}, V^{n+1})$.

As *Hin*(Σ) is finite and $W^{n+1} \subseteq W^n$, then for some $m \geq 0$ this inductive process stops creating new structures. (That is, for all $j \geq m$, $\mathfrak{M}^j = \mathfrak{M}^m$.) Define $\mathfrak{F}$ $(= (W, Q_\pi)_{\pi \in \Pi})$ to be $\mathfrak{F}^m$ and define $\mathfrak{M}$ $(= (\mathfrak{F}, V))$ to be $\mathfrak{M}^m$.

The reader should contrast this use of Hintikka sets with the way we used them in our discussion of PSPACE. The $Witness$ algorithm carefully builds sequences of ever smaller Hintikka sets using only PSPACE resources. In sharp contrast to this, the first step of *Elimination of Hintikka sets* forms all possible Hintikka sets (and there are exponentially many), and subsequent steps filter out the useless ones.

Theorem 6.54 *The satisfiability problem for* **PDL** *is solvable in deterministic exponential time.*

Proof. Given a PDL formula ψ, we will test for its satisfiability as follows. Letting Σ be the set of all ψ's subformulas, we form *Hin*(Σ) and perform elimination of Hintikka sets. This process terminates yielding a model $\mathfrak{M} = (W, Q_\pi, V)_{\pi \in \Pi}$. We will shortly prove the following claim, for all formulas $\phi \in \Sigma$:

$$\phi \text{ is satisfiable iff } \phi \in H \text{ for some } H \in W. \tag{6.16}$$

If we can prove this claim, the theorem follows. To see this, note that the number of Hintikka sets over Σ is exponential in the size of ψ, and the process of constructing $\mathfrak{M}^{n+1}$ out of $\mathfrak{M}^n$ is a deterministic process that can be performed in time polynomial in the size of the model, and hence elimination of Hintikka sets is an EXPTIME algorithm.

So it remains to establish (6.16). For the right to left direction, we will show that if $\phi \in H$ for some $H \in W$, then $\mathfrak{M}$ itself satisfies ϕ at H. Indeed, we will show that for all $\phi \in \neg FL(\Sigma)$ and all $H \in W$, $\mathfrak{M}, H \Vdash \phi$ iff $\phi \in H$. This proof is by induction. The clause for propositional symbols is clear, and the step for boolean combinations follows using clauses (i) and (ii) in the definition of Hintikka sets. For the step involving the modal operators we need the following subclaim:

$$\begin{array}{l}\text{for all } \langle\pi\rangle\chi \in \neg FL(\Sigma),\ \langle\pi\rangle\chi \in H \text{ iff}\\ \text{for some } H' \in W \text{ we have } Q_\pi H H' \text{ and } \chi \in H'.\end{array} \tag{6.17}$$

The left to right direction of (6.17) is immediate from the construction of $\mathfrak{M}$, for at the end of the elimination process only the demand-satisfied Hintikka sets remain. The right to left direction follows by induction on the structure of π; we demonstrate the base case and the step for modalities constructed using $*$. Suppose that for some basic program a there are Hintikka sets H and H' such that HQ_aH' and $\chi \in H'$. As we built the relation Q_a by a sequence of eliminations and restriction, it follows that if HQ_aH' then HQ_a^0H' — and hence it follows by definition that $\langle a\rangle\chi \in H'$. Next, suppose that for some program π^* there are Hintikka sets H and

H' such that $HQ_{\pi^*}H'$ and $\chi \in H'$. But this means there is a finite sequence

$$H = H_0 Q_\pi H_1 \ldots H_{n-1} Q_\pi H_n = H'.$$

As $\chi \in H'$ it follows inductively that $\langle\pi\rangle\chi \in H_{n-1}$ and hence (due the fact that all Hintikka sets are Fisher-Ladner closed) that $\langle\pi^*\rangle\chi \in H_{n-1}$. Again, by induction on π it follows that $\langle\pi\rangle\langle\pi^*\rangle\chi \in H_{n-2}$, whence $\langle\pi^*\rangle\chi \in H_{n-2}$ since this set if Fisher-Ladner closed. By repeating this argument we obtain that $\langle\pi^*\rangle\chi \in H_0 = H$. This establishes the inductive proof of (6.17), which in turn completes the inductive proof of the right to left direction of (6.16).

The fastest way to prove the left to right direction of (6.16) is to make use of ideas developed when proving the completeness of **PDL** in Chapter 4. Recall that we defined $\mathfrak{P}$, the **PDL** model over Σ, to be $(At(\Sigma), \{R^\Sigma_\pi\}_{\pi\in\Pi}, V^\Sigma)$. Here $At(\Sigma)$ is the set of all atoms over Σ, V^Σ is the natural valuation, and R^Σ_π is defined as follows: for any two atoms A and B, and any basic program a, $AR^\Sigma_a B$ holds iff $\widehat{A} \wedge \langle a\rangle\widehat{B}$ is consistent. We defined R_π for arbitrary programs by closing these basic relations under composition, union, and reflexive transitive closure in the usual way.

Now, we first claim that for all programs π, $R^\Sigma_\pi \subseteq Q^0_\pi$. To see this, first observe that as $At(\Sigma) \subseteq Hin(\Sigma)$, all atoms A and B are in W^0. So suppose $AR^\Sigma_a B$. Then, as $\widehat{A} \wedge \langle a\rangle\widehat{B}$ is consistent, by the maximality of Hintikka sets we have that for all $\phi \in B$, if $\langle a\rangle\phi \in \neg FL(\Sigma)$ then $\langle a\rangle\phi \in H$, that is, $AQ^0_\pi B$. Thus for all atomic programs, the desired inclusion holds. But the relations R_π and Q^0_π corresponding to arbitrary programs π are generated out of R_a and Q^0_a in the usual way, hence the inclusion follows for all programs.

The importance of this observation is the following consequence: atoms can never be discarded in the process of elimination of Hintikka sets. This follows from the Existence Lemma for **PDL** (Lemma 4.89, which states that for all atoms A, and all formulas $\langle\pi\rangle\psi \in \neg FL(\Sigma)$, if $\langle\pi\rangle\psi \in A$, there is an atom B such that $AR^\Sigma_\pi B$ and $\psi \in B$. As all atoms belong to W^0, and as $R^\Sigma_\pi \subseteq Q^0_\pi$, it follows that every atom in W^0 is demand-satisfied. Moreover, this demand satisfiability depends only on the presence of other atoms. It follows that Hintikka elimination cannot get rid of atoms; that is, $Hin(\Sigma) \subseteq W$.

But now the left to right direction of (6.16) follows easily. Suppose that ϕ is satisfiable. Then ϕ is **PDL**-consistent, which means it belongs to at least one atom in Σ. This atom will survive the elimination process, and we have the result. $\dashv$

This establishes the result we wanted: an EXPTIME algorithm for deciding the satisfiability problem for PDL. One question may be bothering some readers: what is the relationship between the models $\mathfrak{M}$ and $\mathfrak{P}$ in the proof of Theorem 6.54? Let us consider the matter. In the proof, we observed that all atoms survive the Hintikka elimination process. In fact, only atoms can survive. (To see this, simply observe that if some inconsistent Hintikka set H survived the Hintikka process, then by (6.16), every formula in H would be satisfied in $\mathfrak{M}$ at H. But as $\mathfrak{M}$ is a regular model, this is impossible.) Hence $\mathfrak{M}$, like $\mathfrak{P}$, is a model built over the set of atoms. Moreover, we showed in the course of proving the previous theorem that every relation in $\mathfrak{P}$ is a subrelation of the corresponding relation in $\mathfrak{M}$. It follows that $\mathfrak{P}$ is a submodel of $\mathfrak{M}$.

Actually, we can say a little more. Recall from Exercise 4.8.4 that $\mathfrak{P}$ is isomorphic to a certain filtration. In fact, $\mathfrak{M}$ is isomorphic to a filtration over the same set of sentences. Which filtration? We leave this as an exercise for the reader; see Exercise 6.8.4.

Exercises for Section 6.8

6.8.1 Enrich the basic modal language with the global modality A. (This was defined in Section 6.5.) Show that the satisfiability problem for the enriched language over the class of all frames is EXPTIME-hard.

6.8.2 As in the previous exercise, enrich the basic modal language with the global modality A. Use elimination of Hintikka sets to show that the satisfiability problem for the enriched language over the class of all frames is solvable in EXPTIME.

6.8.3 In this exercise we investigate the complexity of deterministic PDL.

(a) Change the PDL-hardness proof so that it works for deterministic PDL. How many programs do you need? Are two programs sufficient?
(b) Encode with just one functional program that a model has an exponential deep path. Use this to describe n-corridor tiling. What can you conclude?
(c) So by now we might have a suspicion that with only one program, the satisfiability problem for deterministic PDL might be in PSPACE. But how to prove that? The best way is to find a proof in the literature which can be used almost immediately. What are the crucial features of functional PDL with one program? Think of a temporal logic which has precisely these same features. Can you interpret functional PDL into that temporal logic, using some kind of translation function? If so, what is the complexity of that function? What can you conclude?

6.8.4 Determine the exact relationship between the models $\mathfrak{P}$ and $\mathfrak{M}$ discussed following the proof of Theorem 6.54.

6.8.5 PDL has an EXPTIME-complete satisfiability problem. Suppose we add the the universal modality to the language. What is the complexity of the resulting satisfiability problem?

6.9 Summary of Chapter 6

- *Decidability and Undecidability*: A logic is called decidable if its satisfiability problem (or equivalently, its validity problem) is decidable. Otherwise it is called undecidable.

- *Decidability via the Finite Model Property*: While possession of the finite model property does not guarantee decidability, finite models can be used to prove decidability given some extra information about the models or the logic. The decidability of many of the more important modal logics, including **PDL**, can be established using such arguments.
- *Decidability via Interpretations*: Another important technique for establishing decidability is via interpretation in decidable logical theories, most notably the monadic second-order theories of countable finitely- or ω-branching trees. If a modal logic is complete with respect to a class of models that can be viewed as monadic second-order definable substructures of such a tree, its decidability follows.
- *Quasi-Models and Mosaics*: Even when a modal logic lacks the finite model property, it is sometimes possible to prove decidability using finite *representations* of the information contained in satisfying models. Quasi-models and mosaics are such representations.
- *Undecidability*: Undecidability arises easily in modal logic. Moreover, not all undecidable modal logics have the simplest degree of undecidability; many are highly undecidable.
- *Tiling Problems*: Tiling problems can be used to classify the difficulty of both decidable and undecidable problems. The simple geometric ideas underlying them makes them a useful tool for investigating modal satisfiability problems.
- *The Modal Significance of NP*: Only modal logics with the polysize model property with respect to particularly simple classes of structures can be expected to have satisfiability problems in NP. Some important logics, such as the normal logics extending **S4.3**, fall into this category.
- *The Modal Significance of PSPACE*: Assuming that PSPACE $\neq$ NP, most modal satisfiability problems are *not* solvable in NP, but are at least PSPACE-hard. For example, every normal logic between **K** and **S4** has a PSPACE-hard satisfiability problem. Explicit PSPACE algorithms are known for some of these logics.
- *The Modal Significance of EXPTIME*: Modal languages containing a modality $\langle r \rangle$ and a matching reflexive transitive closure modality $\langle r^* \rangle$ often have EXPTIME-hard satisfiability problems. The two person corridor tiling game is an attractive tool for proving modal EXPTIME-hardness results, and elimination of Hintikka sets is a standard way of defining EXPTIME algorithms.

Notes

Finite models have long been used to establish decidability, both in modal logic and elsewhere. Arguments based on *finite* axiomatizability together with the f.m.p. are widely used (Theorem 6.15); this approach traces back to Harrop [219]. Also popular is the use of the *strong* finite model property; our formulation (Theorem 6.7) is based on Goldblatt's [183]. The fact that a *recursive* axiomatization together with the f.m.p. with respect to a *recursively enumerable* class of models guarantees decidability (Theorem 6.13) seems to have first been made explicit in Urquhart [432]. The main point of Urquhart's article is to prove the result we presented as Exercise 6.2.5: there is a normal modal logic which is recursively axiomatizable, and has the f.m.p., but is undecidable. This shows that the use of finite axiomatizations in the statement of Theorem 6.15 cannot be replaced by recursive axiomatizations, and Urquhart states Theorem 6.13 as the correct generalization. Exercise 6.2.4 is due to Hemaspaandra (*née* Spaan); see Spaan [412]. For Craig's Lemma, see Craig [95].

The original proof of Rabin's Tree Theorem may be found in Rabin [372]. Rabin shows that the decidability of $\mathrm{S}n\mathrm{S}$ for $n > 2$ or $n = \omega$ is reducible to the decidability of S2S, and the bulk of his paper is devoted to proving that S2S is decidable. Rabin's paper is demanding, and simpler proofs have subsequently been found; for an up to date survey of Rabin's Theorem and related material, see Gecseg and Steinby [174] and Thomas [424]. Rabin's Theorem was applied in modal logic almost immediately: Fine [135] used it to prove decidability results in second-order modal logic (that is, modal logic in which it is possible to bind propositional variables), and Gabbay [154, 155, 156] applied it to a wide range of modal logics in many different languages. Gabbay, Hodkinson, and Reynolds [163] is a valuable source on the subject.

Two kinds of variations on Rabin's Tree Theorem are relevant to our readers. First, the *weak* monadic second-order theory of n successor functions ($\mathrm{WS}n\mathrm{S}$) constrains the set variables to range over finite sets only. The decidability of $\mathrm{WS}n\mathrm{S}$ — which is due to Thatcher and Wright [421] and Doner [121] — is based on a close correspondence between formulae in $\mathrm{WS}n\mathrm{S}$ and finite automata; any relation ϕ definable in $\mathrm{WS2S}$ can also be defined by a tree automaton A_ϕ that encodes the satisfying assignments to the formula in the labels on the nodes of the tree that it accepts. The MONA system [226] implements this decision procedure. Despite the non-elementary worst-case complexity of $\mathrm{WS2S}$, MONA works well in practice on a large range of problems; Basin and Klarlund [28] offer empirical evidence and an analysis of why this is the case. At the time of writing there are no experimental results evaluating the performance of tools such as MONA on logics such as propositional dynamic logic. Muller *et al.* [344] use reductions to $\mathrm{WS2S}$ to explain why many temporal and dynamic logics are decidable in EXPTIME.

A second variation is important when working with expressive modal languages (for example, those containing the until operator U) over highly restricted classes of models (for example, models isomorphic to the real numbers in their usual order) it may be necessary to appeal to stronger results about specific classes of structures;

Burgess and Gurevich [78] and Gurevich and Shelah [207] are essential reading here.

Prenex normal form fragments of first-order logic are defined using strings over $\{\exists, \exists^*, \forall, \forall^*\}$; for instance, $\exists\forall^*$ represents the class of first-order formulas in prenex normal form where the quantifier prefix starts with an existential quantifier and is followed by a (possibly empty) sequence of universal quantifiers. The decidability of prenex normal form fragments seems to have been studied at least since early 1920s, which is when Skolem showed that $\forall^*\exists^*$ is undecidable. In 1928, Bernays and Schönfinkel gave a decision procedure for the satisfiability of $\exists^*\forall\exists^*$ sentences. Gödel, Kalmár and Schütte, independently in 1931, 1933 and 1934 respectively, discovered decision procedures for the satisfiability of $\exists^*\forall^2\exists^*$ sentences. In 1933, Gödel showed that $\forall^3\exists^*$ sentences form a reduction class for satisfiability. More recently, Kahr in 1962 proved the undecidability of $\forall\exists\forall$. Consult Börger *et al.* [69] for references and an encyclopedic account of prenex normal form fragments. For recent work on the relevance of such fragments to modal logic, see Hustadt [243].

That the two-variable fragment of any first-order language is decidable is relevant to a number of modal decidability problems. The first decidability result for this fragment (without equality) was obtained by Scott [393]; Mortimer [343] established decidability of the two-variable fragment with equality. In contrast, for $k \geq 3$, the k-variable fragment is undecidable. Consult Grädel, Kolaitis, and Vardi [201] for complexity results, and Grädel, Otto, and Rosen [202] for related results.

But perhaps the most natural way to reduce a modal logic is — to another modal logic! Such reductions are far likelier to yield not only decidability results, but information about complexity as well. Embeddings of temporal logic into the basic modal language were first studied by Thomason in the mid 1970s (see, for example, [429]). The approach has gained a new lease of life recently — important results on the approach can be found in Kracht and Wolter [289] and Kracht [286].

Our use of quasi-models and mosaics has it roots in the work of Zakharyaschev and others. In particular, Zakharyaschev and Alekseev [460] use such arguments to show that all finitely axiomatizable normal logics extending **K4.3** are decidable, and Wolter [450] uses them to show that all finitely axiomatizable tense logics extending $\mathrm{K_t}4.\mathbf{3}$ are decidable too.

The mosaic method for proving decidability of a logic stems from Németi [345] who proved that various classes of relativized cylindric algebras have a decidable equational theory. It has since been used for a wide range of logics, often with a multi-dimensional flavor; see for instance Marx and Venema [326], Mikulás [335], Reynolds [379], Wolter and Zakharyaschev [454], Wolter [453], or the references in our Notes on the guarded fragment in Chapter 7. With hindsight, even Gödel's proof of the decidability of the satisfiability problem for the $\forall^2\exists^*$ prenex sentences can be called a mosaic style proof as well; see the very clear exposition in the monograph [69]. Mosaics can also be used to investigate modal complexity theory; see Marx [322] for further details.

Constructing specific examples of undecidable modal logics is not trivial, and Thomason [427] contains the earliest explicit example of an undecidable normal logic in the basic modal language that we know of. Undecidable logics can be constructed in a variety of ways. Urquhart's [432] definition of Λ_U (see Exercise 6.2.5) is neat, if abstract. For undecidable logics in the basic modal language constructed by detailed simulation of a concrete model of computation (namely, Minsky machines), see Chagrov and Zakharyaschev [86, Chapter 16].

We have chosen to focus on tiling problems (or domino problems, as they are sometimes called). These were introduced in Wang [446] and have since been used in a variety of forms to prove undecidability and complexity results. Proofs that the $\mathbb{N} \times \mathbb{N}$ tiling problem is undecidable can be found in Berger [50], Robinson [381], and Lewis and Papadimitriou [308]. Two important papers on tiling are Harel [216, 217]: these demonstrate the flexibility of the method as a tool for measuring the complexity of logics. Harel uses tiling to give an intuitive account of highly undecidable (and in particular, Σ^1_1-complete) problems, and these two papers are probably the best starting point for readers interested in learning more. The logic KR used in the text to illustrate the tiling method is a notational variant of Kasper Rounds logic, which is used in computational linguistics to analyze the notion of feature structure unification. Decidability and complexity results for (various versions of) Kasper Rounds logic can be found in Kasper and Rounds [271] and Blackburn and Spaan [59]; the latter is the source for Theorems 6.31 and 6.34. A wide range of related results can be found in the literature (see for example Harel [215], Halpern and Vardi [209], and Passy and Tinchev [362]). Even in quite modest languages, asserting something about all paths through a model can lead to extremely high complexity; for a deeper understanding of why this is so, we refer the reader to Harel [216, 217], and to Harel, Kozen and Tiuryn [218].

As to complexity-theoretic classifications of modal satisfiability and validity problems, Ladner [299] is one of the earliest analyses; this classic paper is the source of Ladner's Theorem and much else besides — it is required reading! Halpern and Moses [212] is an excellent introduction to the decidability and complexity of multi-modal languages. We strongly recommend this article to our readers — especially those who are encountering complexity theoretic ideas for the first time.

But to return to the results in this chapter, the NP-completeness of **S5** was proved in Ladner [299]. Ono and Nakamura [353] is the source of Theorem 6.38; in that paper it is also shown that the complexity of the satisfiability problems in the language with F and P with respect to the following flows of time are all NP-complete: linear transitive flows of time without endpoints, and dense linear transitive flows of time without endpoints (see Exercise 6.6.3). Hemaspaandra's Theorem, that all normal modal logics extending **S4.3** are NP-complete, may be found in Spaan [412] and Hemaspaandra [221]). As an aside, the satisfiability problem for the flow of time $(\mathbb{N}, \leq)$ in the language with just F was shown to be NP-complete by Sistla and Clarke [408]; the satisfiability problem is also shown to be NP-complete for formulas using F and the so-called operator nexttime operator. NP-complete modal-like logics were also investigated in the area of description logic; see below for references. Many NP-completeness results make use of Lemma 6.36, that frame membership is decidable in polynomial time for first-order definable frame classes (see in Exercise 6.6.1). This is a standard result in finite model theory, and you can find a proof in Ebbinghaus and Flum [126].

The key results on PSPACE come from Ladner [299]. Ladner first establishes the existence of PSPACE algorithms for **K**, **T**, and **S4**. His proof of the PSPACE-completeness of **K** is like that given in the text, save that Ladner uses 'concrete tableaux' (that is, his algorithm specifies how to construct the required atoms) rather than 'abstract tableaux' (which factor out the required boolean reasoning). Concrete tableaux are also used by Halpern and Moses [212] to construct PSPACE algorithms for multi-modal versions of **K**, **S4** — and indeed **S5**; as they show, logics containing two **S5** modalities are PSPACE-hard. This paper gives a very clear exposition of how to use tableaux systems to establish decidability and complexity results. The abstract tableaux systems used in this chapter are based on the work of Hemaspaandra [413, 412, 221]. In the description logic community, tableaux systems are often called *constraint systems* [123]; *description* logics (also known as *concept* languages or *terminological* logics) are essentially multi-modal languages, often equipped with additional operators to facilitate the representation of knowledge, with global constraints (the so-called TBox), or with means to reason about individuals and properties (the so-called ABox). Unlike the modal logic community, in the description logic community considerable attention has been paid to reasoning tasks other than satisfiability or validity checking, such as subsumption checking, instance checking, and reasoning in the presence of a background theory [122].

In the text (page 403) we also mentioned the fact that, over the natural numbers (with $<$ and the successor function S), the temporal logic with the until operator has a PSPACE-complete satisfiability problem; this result is due to Sistla and Clarke [408]. In the same paper, the authors also show that the satisfiability problem for $(\mathbb{N}, <)$ is PSPACE-complete for each of the following systems: F and X; U (until); U, S (since), X; and the extended temporal logic ETL due to Wolper [449].

The effect of bounding the number of proposition letters and the degree of modal formulas has been studied by Halpern [210]. In addition to the results mentioned in Exercises 6.6.5 and 6.7.7, he shows that the PSPACE-completeness results of

Ladner and Halpern and Moses hold for multi-modal versions of **K**, **T**, **S4**, **S5**, even if there is only one proposition letter in the language. If we restrict to a finite degree, then the satisfiability problem is NP-complete for all the logics considered, but **S4**, and if we impose both restrictions, the complexity goes down to linear time in all cases.

The EXPTIME-hardness of **PDL** (Theorem 6.52) is due to Fisher and Ladner [143], who explicitly construct a PDL formula which simulates the actions of a linear space bounded space bounded alternating Turing machine. The (simpler) proof given in the text stems from Chlebus [91], which establishes the EXPTIME hardness of the two person corridor tiling game (via a reduction from alternating Turing machines) and uses it to provide a new proof of EXPTIME hardness for **PDL**. Another proof of this via two person corridor tiling can be found in Van Emde Boas [128], and we have also drawn on this; the recursive formulation of the game halting condition is due to Maarten Marx.

The existence of an EXPTIME algorithm for **PDL**, and the method of eliminating Hintikka sets, comes from Pratt [366]. Other applications of the method can be found in multi-modal logics of knowledge equipped with a common knowledge operator (see Halpern and Vardi [209], or Fagin *et al.* [133]); in computational tree logic (CTL; see Emerson [129]); in expressive description logics (see Donini *et al.* [123]); and in work on the global modality (see Marx [322] or Spaan [412]).

One important approach to the analysis of modal complexity has not been discussed in this chapter: the use of finite automata. The theory of automata has been a subject of research since the 1960s (Büchi [70], Thatcher and Wright [421], Rabin [372]). Especially relevant to temporal and dynamic logics has been a resurgence of interest in finite automata on infinite objects in the 1980s and 1990s; see Gecseg and Steinby [174], Hayashi [220], and Thomas [423, 424]. A wide variety of automata have been studied, and complexity results for their acceptance problems are known. It is often possible to analyze the complexity of modal satisfiability problems by reducing them to acceptance problems for types of automata. For example, general automata-theoretic techniques for reasoning about relatively simple logics using Büchi tree automata have been described by Vardi and Wolper [435].

We conclude on a more general note. In this chapter we have focussed mainly on satisfiability and validity problems — what about the decidability and complexity of other reasoning tasks? For a start, the *global* satisfiability problem (whether there is a model which satisfies a formula at *all* its points) is important in many applications and quite different from the (local) satisfiability problem discussed here. The discussion of the global modality in Section 6.5 and Exercise 6.8.1 has given the reader some of the flavor of such problems; for more, see Marx [322]. Other reasoning tasks that are closely related to the *global* satisfiability problem, are often studied in the area of description logic mentioned before; see De Giacomo [102] or Areces and de Rijke [15].

Furthermore, there is a great deal of interest in building practical systems that evaluate formulas (not necessarily modal ones) in models; this field is known as *model checking*. Many interesting problems can be usefully viewed as model checking problems, and representations which enable evaluation to be performed efficiently — even when the models contain a very large number of states — have been developed. For an intuitive, modally oriented, introduction to the basic ideas, see Halpern and Vardi [213]. For further pointers to the model checking literature, see [332, 93, 245].

Third, it is interesting to inquire into the decidability or otherwise of a wide range of metalogical properties of logics. One such result was mentioned in Section 3.7: Chagrova's Theorem tells us that it is undecidable whether a first-order property of frames can be defined by a modal formula. And many other questions along these lines can be raised (for example: is it decidable whether a new proof rule is admissible in a given logic?). The best sources for further information on such topics are Chagrov and Zakharyaschev [86, Chapters 16 and 17] and Kracht [286]. Another line of results that we should mention here is work on the following question: given two (finite) models $\mathfrak{M}$ and $\mathfrak{N}$, how hard is it to decide whether they are bisimilar? Ponse *et al.* [364] contains a number of valuable starting points for such questions.