CHAPTER 21

Encoding Two-Valued Nonclassical Logics in Classical Logic

Hans Jürgen Ohlbach

Andreas Nonnengart

Maarten de Rijke

Dov M. Gabbay

SECOND READERS: Nikolaj Bjorner.

Contents

HANDBOOK OF AUTOMATED REASONING
Edited by Alan Robinson and Andrei Voronkov

1. Introduction

Classical logic has been the principal logic used in applications for many years. One of its main application areas has been mathematics, simply because it is ideally suited for the formalization of mathematics. Its model theory and proof theory have had an enormous impact on the foundations of mathematics. Many logicians continue to hold the view that logic has four subareas: model theory, proof theory, recursion theory and set theory. In all these four areas classical logic reigns supreme.

But as soon as we try to apply logic to the more human-oriented applications arising in computer science, artificial intelligence, natural language processing and philosophy, we find that we need to depart from the basic features of classical logic. These departures come in several kinds:

1. extend the language of classical logic with additional operators, connectives, and quantifiers to service applications involving time, knowledge, belief, necessity, actions, etc.;
2. restrict the language to guarantee better computational or logical properties, such as (strong forms of) decidability and interpolation;
3. change the basic deductive structure of classical logic to create new logics (intuitionistic, relevance, linear, paraconsistent, many-valued, fuzzy, etc.) to obtain systems whose reasoning mechanisms are closer to the domain being modeled.

For many applications a combined system may need to be used, for example systems like intuitionistic many-valued logics, or a fuzzy modal logic, or relevance modal logic etc. At the same time, under pressure from application areas, the notion of a logic itself has changed; the basic datastructures and concepts of "logic" have evolved. A theory need no longer be a set of formulae, but it can be a list, or a list of lists, a tree, or a labeled datastructure in general. Inference mechanisms are defined on top of these structures. We witness a wide landscape of nonmonotonic logics (default logic, defeasible logic, circumscription, etc.) and inference mechanisms (negation as failure, abduction, defaults, etc.). Today, a logical system may be a combination of many of the above and may look very different from classical logic (see [Gabbay 1996]).

In this chapter we are concerned with the more traditional monotone two-valued nonclassical logics, and modal logic is the most prominent example of these logics. Despite this restriction, we are still faced with an overwhelming number of logics. Encoding these logics into other logics gives us a powerful tool for understanding them from both a logical, algorithmic, and computational point of view, as we will see below. Before we come to the main issues of the chapter, we will explain the main topics being discussed in the context of nonclassical logics; this will enable us to explain what this chapter is about, and what it is not about.

1.1. *Expressive power*

The two most important properties of a logic are its *expressive power* and the (decidability of) the *reasoning tasks* that can be performed in the logic, in particular theoremhood. In this subsection we focus on expressive power, leaving reasoning tasks to the next one.

The expressive power of a logic may be analyzed from a number of angles. First of all, how do we express things in a given logic, or put differently, what is the *syntax* of the logic? This may seem a trivial point, but a poor syntax may be an important barrier to using a given logic.

Second, *what* do we say in a given logic? That is, what is the *meaning* of our well formed formulae (wffs)? This is a non-trivial question, because it is application dependent and because it requires a thorough understanding of the intended application. For example, if one wants to develop a logic of *knowledge and belief* [Fagin, Halpern, Moses and Vardi 1995], one has to find a precise characterization of 'knowledge' and 'belief,' before one can start specifying the corresponding logic.

There are various ways of specifying the meaning of wffs. Usually, this is done either *proof theoretically*, *semantically* or by *embedding* the logic into an already established logic. All of these methods specify the meaning of formulae in a rather indirect way. A proof theoretic specification by means of a *Hilbert system*, for example, is a kind of grammar for generating *theorems*. The theorems are a distinguished subset of the well formed formulae, which are supposed to 'be true' a-priori, without any extra assumptions.

A *semantic* characterization of the meaning of a wff is a mapping of the syntactic elements to a mathematical domain. The latter are usually simple and well understood, for example sets equipped with functions and relations. Thus, the very idea of a semantic characterization is to explain a new logical system in terms of something old, simple and well understood.

This familiar system actually need not be a mathematical domain. It can also be a previously defined logic. If the wffs of the new logic are translated into the wffs of a given logic, we speak of *embedding* or *translating* the new logic into the given one. In this way we can use an old logic to explain the meaning and expressive power of the wffs of the new logic.

Third is the issue of *comparing* the relative expressive power of logics. As indicated at the start of this section, we often need to depart from a given logic to cater for the needs of new applications. As a concrete example, one can think of the area of *description logics* (see [Calvanese et al. 2001], Chapter 23 of this Handbook), where lots of logical languages have been explored with varying sets of admissible connectives and quantifiers. Some versions allow negation to occur only in front of predicate symbols, which makes them different from logics where negation is allowed to occur everywhere. Others impose restrictions on the arguments of some of the quantifiers. The resulting plethora of logics creates many theoretical challenges; two of the most interesting are

- identifying 'dangerous' and 'safe' constructs in syntax and semantics, which do or do not make the decision problem harder when present;
- developing tools for comparing the relative expressive power of logics, in terms of special semantic or algorithmic characteristics;

As we will see below, encoding logics into each other or in some 'big' background logic, often provides a powerful way of addressing the above challenges.

1.2. Reasoning tasks

In addition to its expressive power, the other important feature of a logic are its *reasoning tasks*, and the *methods* for performing these. Reasoning tasks that have traditionally been considered include consequence or theoremhood proving, counterexample generation, model checking, subsumption checking, and satisfiability checking, see [Clarke and Schlingloff 2001, Calvanese et al. 2001] (Chapters 24 and 23 of this Handbook) for examples and details). To simplify our presentation, let us focus on the first of these.

A first issue to be addressed when studying the problem of consequence checking for a given logic is *decidability* of the problem; and if theoremhood is decidable, what is its complexity? This goes along with the development of decision or — if theoremhood is not decidable — semi-decision proof procedures.

An actual proof procedure has several layers. The lowest layer consists of the basic *calculus*, a set of non-deterministic inference rules. Soundness (the rules identify only theorems) and completeness (the rules reject any non-theorem) of the calculus ensures that it works as expected. Since the rules of the basic calculus are non-deterministic, they generate a *search space*. The overall efficiency of a proof procedure depends on how this search space is explored. The next layer in a proof procedure therefore consists of *search strategies*. Search strategies, for example ordering strategies or selection strategies in predicate logic, prune the search space by restricting the application of the rules; see [Bachmair and Ganzinger 2001] (Chapter 2 of this Handbook).

Since search strategies cut out large parts of the search space, additional completeness proofs are called for. The resulting system is still non-deterministic. Therefore, in a third layer *search heuristics* are employed to control the application of the inference rules.

Orthogonal to these three layers is another component of a good proof procedure: the ability to recognize and eliminate redundancies in the search space. Inference steps which obviously cannot contribute to the final solution should be avoided as far as possible. Tautology and subsumption deletion in predicate logic resolution systems provide typical examples; again, see [Bachmair and Ganzinger 2001] (Chapter 2 of this Handbook).

Most successful implementations of proof procedures do not only rely on the strength of the calculus, the strategies, heuristics and redundancy elimination in the system, but also on sophisticated technical optimizations on various levels. This ranges from good datastructures and strong indexing techniques for dealing with large sets of terms and formulae [Graf 1996] to parallel processing on computer networks.

Let us return to a theme introduced at the start of this section (page 1405). Applications of logic call for many new logics to be defined. Changing the specification of a logic influences most of its key properties, in particular the decidability of theoremhood, the complexity of the decision problem, the structure of the proof procedures etc. More concretely, the following have proved to be the key concerns:

- identifying 'dangerous' constructs in syntax and semantics, which make the decision problem harder when present;
- understanding why certain modifications do not influence the complexity of the decision procedure;
- developing parametrized proof procedures, which work for a whole class of logics by changing the parameters;
- developing proof procedures for more general logics which automatically become efficient proof procedures for less general logics when faced with a problem in the less general logic.

As should be clear from the above discussion, proof procedures have many different aspects to them; and developing procedures for a given logic is a non-trivial task.

1.3. Why encode?

In the previous subsections we have discussed what we take to be the main aspects of logics, classical or nonclassical, viz. their expressive power and their reasoning tasks. With the advent of more and more special purpose logics, we are faced with the daunting task of having to analyze the expressive power and develop proof procedures for each of these logics. This is where we think that methods of encoding one logic (the *source logic*) into another logic (the *target logic*) may provide an invaluable tool — quite often they are the only options available.

In particular, methods for encoding a source logic into a target logic can be used to achieve the following:

1. To grasp the expressive power, both of the source logic and of the target logic.
2. To export any proof procedures and tools that we may have for the target logic to the source logic.
3. To grasp the computational costs of various reasoning tasks, again both for the source logic and for the target logic.
4. To combine two logics, sometimes the most natural solution is to translate both into the same target logic, and to work inside the target logic (see [Gabbay 1999]).

In the remainder of this chapter we elaborate on these ideas, especially the first two, and present numerous uses and applications.

The automated reasoning community is by no means the first to think of encoding one logic into another as a useful tool. In mathematical logic, the topic of *interpreting* one logic inside another one has a long history, and it has in fact been motivated by much the same motivation as we have given above (see, for instance, [Hájek and Pudlák 1991]). In a similar spirit, philosophers of science have long been interested in *reducing* systems — formal or informal — to each other with a view to understanding what claims scientific theories are actually making about real world phenomena [Quine 1953].

But let us return to automated reasoning matters. We want to raise an important issue concerning the encoding of logics that some people have found troubling. We will mostly be encoding 'small' decidable source logics into 'big' undecidable target logics, and we will perform reasoning tasks for the source logic within the target logic, thereby 'reducing' a decidable problem to an undecidable one. Surely, it must be much more effective to reason directly within the source logic, say with the techniques explained in [Calvanese et al. 2001] (Chapter 23 on description logics in this Handbook) or [Waaler 2001] (Chapter 22 on tableaux methods for nonclassical logics)? A number of replies are appropriate. For getting a really efficient system for a given source logic, one has to exploit the characteristics of its (translated) formulae as much as possible, and this usually requires special implementations of any tools that one may have for the target logic. At the time of writing all systematic comparisons between proof procedures working directly on the original formulae on the one hand, and proof procedures working at the translated predicate logic formulae on the other hand have been done using general purpose predicate logic theorem provers. Predicate logic theorem provers which are highly optimized for the particular fragment of predicate logic which is the target of the translation are not available. Nevertheless, it turned out that the general purpose theorem provers can compete well with highly optimized tableaux provers for the source logics [Hustadt 1999]. Since all these theorem provers are improving every day, the comparisons are only snapshots that may be outdated very rapidly; therefore it is not worth reporting details of the experiments in this chapter.

We think that encoding nonclassical logics into first-order logic is important, not just for understanding the expressive power and computational powers of the source logics, but also because it gives users of nonclassical logics access to the sophisticated, state-of-the-art tools that are available and that continue to be developed in the area of first-order theorem proving. At present, no purpose built theorem prover for nonclassical logic has the level of sophistication that first-order provers have. Admittedly, a pre-processing step or a very careful encoding is often needed to exploit the restricted expressive power of the source logic and to help a general purpose tool become an efficient system for a given source logic, but even without this, an encoding gives direct access to valuable tools.

1.4. What this chapter is about

This chapter is about methods for encoding (axiomatizing, translating, simulating) a nonclassical logic in predicate logic, in particular in first-order predicate logic. We will focus mainly on matters related to expressive power and reasoning tasks; specific questions that we will be concerned with are:

1. What are the different options for encoding the formulae in predicate logic?
2. Do particular translation methods work for a whole class of logics, and how do the differences manifest themselves in the encoding?
3. What is the fragment of predicate logic, into which the formulae of a given logic or a whole class of logics are translated with a particular translation method?

4. What kind of proof procedures are adequate for the translated formulae?
5. Can we exploit special features of the source logic to enhance the encoding?
6. Are there general purpose proof procedures for predicate logic, which work well enough for the translated formulae?

In general, the answers to these questions depend on the particular translation method being used, and we will see that many of the questions are still open.

In this chapter we take for granted that the enterprise of encoding nonclassical logics into classical logic has definite merits. We will concentrate on the fundamental ideas and results about encodings. The main ideas of the most important encoding techniques are presented in such a way that after studying the method, the reader should be able to find a suitable encoding for a given new logic herself.

More specifically, this chapter is organized as follows. Section 2 contains basic definitions and may be skipped on first reading. Section 3 explores the first of a number of ways of encoding a source logic into a target logic: syntactic encodings; we discuss the basic ideas and consider the merits of the approach from an automated reasoning point of view. In Section 4 we consider the standard relational translation; in this way of encoding we transcribe the semantics of the source logic in the target logic; we discuss the implications of this approach for our understanding of the expressive power of the source logic, and we discuss how the approach necessitates different ways of encoding if one is to do efficient automated reasoning for the source logic. Some of these alternatives are explored in Sections 5 and 6 where we consider the functional, the optimized functional, and the semi-functional translation. Further variations on and extensions of these ideas are discussed in Section 7. Finally, in Section 8 we formulate some conclusions and open questions.

2. Background

The purpose of this subsection is to set the scene for the remainder of the chapter, by introducing the basic syntactic and semantic prerequisites. We refer the reader to [Blackburn, de Rijke and Venema 2001] for further details.

2.1. Syntax

We will assume basic familiarity with first-order logic (FO). Throughout the chapter we consider a number of nonclassical logics; below we list most of them.

2.1. Definition *(Modal Logic).* Let $\mathcal{P}$ be a collection of proposition letters. The syntax of propositional *(uni-) modal logic* is given by the following rule

$$\mathrm{ML} ::= \mathcal{P} \mid \neg\mathrm{ML} \mid \mathrm{ML} \wedge \mathrm{ML} \mid \Diamond\mathrm{ML} \mid \Box\mathrm{ML}.$$

We freely use the usual boolean abbreviations. The $\Diamond$-operator is usually referred to as 'diamond,' and the $\Box$ as 'box.'

The language of *multi-modal logic* is just like the language of uni-modal logic except that in multi-modal logic we have a collection of diamonds $\langle a \rangle$ and boxes $[a]$, where a is taken from some index set. In the setting of description logics, multi-modal logic is sometimes referred to as $\mathcal{ALC}$; see [Calvanese et al. 2001] (Chapter 23 of this Handbook).

Propositional dynamic logic arises as a special case of multi-modal logic, where the diamonds and boxes are equipped with structure, corresponding to certain program forming operators: ; for sequential composition, $\cup$ for non-deterministic choice, * for finite iterations, and ? for testing. Let $\mathcal{A}$ be a collection of atomic programs; from these we define a collection of complex programs as follows:

$$\mathcal{R} ::= \mathcal{A} \mid \mathcal{R} \,;\, \mathcal{R} \mid \mathcal{R} \cup \mathcal{R} \mid \mathcal{R}^* \mid \text{PDL}?$$

By mutual recursion, PDL formulae are defined by

$$\text{PDL} ::= \mathcal{P} \mid \neg\text{PDL} \mid \text{PDL} \wedge \text{PDL} \mid \langle \mathcal{R} \rangle \text{PDL} \mid [\mathcal{R}]\text{PDL}.$$

Here, $\langle \alpha \rangle A$ is read as 'there exists a terminating execution of the program α which leads to a state where A holds,' and $[\alpha]A$ means that all terminating executions of α lead to a state where A holds. Harel [1984] contains lots of background material on (propositional) dynamic logic. In the setting of description logics, propositional dynamic logic is sometimes referred to as $\mathcal{ALC}_{reg}$; again, see [Calvanese et al. 2001] (Chapter 23 of this Handbook).

Consider a multi-modal language with two diamonds, $\langle F \rangle$ and $\langle P \rangle$. Temporal logic arises when we assign very specific readings to $\langle F \rangle A$ and $\langle P \rangle A$: 'sometime in the future, A will be the case' and 'sometime in the past, A was the case.'

2.2. DEFINITION *(Temporal Logic).* The language of *basic temporal logic* is given by the following rule

$$\text{BTL} ::= \mathcal{P} \mid \neg\text{BTL} \mid \text{BTL} \wedge \text{BTL} \mid \langle F \rangle\text{BTL} \mid [F]\text{BTL} \mid \langle P \rangle\text{BTL} \mid [P]\text{BTL}.$$

Here, $[F]$ ('it will always be the case that') and $[P]$ ('it has always been the case that') are dual operators for $\langle F \rangle$ and $\langle P \rangle$, respectively, just like $\Box$ is the dual of $\Diamond$.

Linear time temporal logic (LTL) extends basic temporal logic by the addition of operators to talk about 'the next moment in time' and 'the previous moment in time,' and operators to express properties of intervals between two moments.

$$\begin{aligned} \text{LTL} \quad ::= \quad & \mathcal{P} \mid \neg\text{LTL} \mid \text{LTL} \wedge \text{LTL} \mid \bigcirc\text{LTL} \mid \Box\text{LTL} \mid U(\text{LTL}, \text{LTL}) \mid \\ & \bullet\text{LTL} \mid \blacksquare\text{LTL} \mid S(\text{LTL}, \text{LTL}). \end{aligned}$$

Here, $\bigcirc$ is the 'next time' operator, and $\bullet$ is the 'last time' operator; $\Box A$ stands for 'always in the future A,' while $\blacksquare A$ stands for 'always in the past A;' and the until-operator $U(A, B)$ is meant to capture that A holds at some point in the future, and until then, B holds; the since-operator S is its 'backward' looking counterpart.

2.3. Definition *(Intuitionistic and Relevance Logic). Propositional intuitionistic logic* (IL) has the same connectives as classical propositional logic, but the implication is weaker than the classical implication.

$$\mathrm{IL} ::= \mathcal{P} \mid \bot \mid \neg\mathrm{IL} \mid \mathrm{IL} \vee \mathrm{IL} \mid \mathrm{IL} \wedge \mathrm{IL} \mid \mathrm{IL} \to \mathrm{IL}.$$

Relevance logic is essentially like intuitionistic logic, but with an even weaker implication. The idea is that $A \to B$ can only be a valid statement if A must actually be used to conclude B, i.e., A is *relevant* for B.

2.4. Definition *(Quantified Modal Logic).* Quantified modal logic is an extension of FO in the same way as propositional modal logic is an extension of propositional logic. Besides all the syntactic elements of FO, there are the two usual modal operators $\Box$ and $\Diamond$.

We now turn to proof-theoretic matters. We first give some general definitions, and then present derivation systems for each of the logics we have just introduced.

2.5. Definition. A *consequence relation* $\vdash$ is a relation between two sets (multisets, lists, trees or any other suitable datastructure) of formulae. $\phi \vdash \psi$ expresses that the set (multiset, list, ...) ψ of formulae is a consequence of the set (multiset, ...) of formulae ϕ. A specification of a logic by means of a consequence relation consists of a set of *axioms*

$$\phi_1 \vdash \psi_1, \phi_2 \vdash \psi_2, \ldots$$

and a set of *inference rules*

$$\frac{\phi_1 \vdash \psi_1, \phi_2 \vdash \psi_2, \ldots}{\phi \vdash \psi.}$$

The axioms are a kind of *a-priori* consequences. For many logics, finitely many axioms are sufficient, but nothing stops us from requiring infinitely many axioms. The inference rules allow us to derive new (*output*) consequences from a finite or infinite set of given (*input*) consequences. Again, there may be finitely many or infinitely many inference rules, but in most cases this number is very small. The inference rules may have side conditions restricting there applicability with extra requirements on the structure of the formulae involved.

More generally, the inference rules may require the non-derivability of some consequences. This means that some of the expressions $\phi_i \vdash \psi_i$ in the input part of the inference rule may actually occur in the form *not* $\phi_i \vdash \psi_i$. This gives the system a nonmonotonic behavior because adding *more* axioms may cause *fewer* derived consequences.

2.6. Definition. The consequence relation $\vdash \psi$ of a *Hilbert system* (HS) has an empty set of premises and just a single formula as conclusion. $\vdash \psi$ means that ψ is a theorem.

propositional part	$\left\{\begin{array}{l} \vdash A \to (B \to A) \\ \vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C)) \\ \vdash ((A \to B) \to A) \to A \\ \vdash A \wedge B \to A \\ \vdash A \wedge B \to B \\ \vdash A \to (B \to A \wedge B) \\ \vdash A \to (A \vee B) \\ \vdash B \to (A \vee B) \\ \vdash (A \to C) \to ((B \to C) \to (A \vee B \to C)) \\ \vdash (A \to \neg B) \to (B \to \neg A) \\ \vdash \neg\neg A \to A \end{array}\right.$	(2.1)
K	$\vdash \Box(A \to B) \to (\Box A \to \Box B)$	(2.2)
modus ponens	$\dfrac{\vdash A \qquad \vdash A \to B}{\vdash B}$	(2.3)
necessitation rule	$\dfrac{\vdash A}{\vdash \Box A}$	(2.4)

Table 1: Axioms and rules for **K**

2.7. Definition *(*$\mathbf{K}$*,* $\mathbf{K}_m$ *and* **PDL***).* The *basic (uni-) modal logic* is called **K** (after Kripke), and it is axiomatized by the axioms and rules listed in Table 1.

The diamond operator $\Diamond$ is defined in terms of the box operator $\Box$ by $\Diamond p \leftrightarrow \neg\Box\neg p$.

The set of axioms for the *basic multi-modal logic* $\mathbf{K}_m$ (where $m \in \mathbb{N}$) consists of (2.1), m copies of the K axiom (2.2) and the necessitation rule, one for each modal operator $[\alpha]$, and modus ponens (2.3).

An axiom system **PDL** for propositional dynamic logic is obtained by taking the **K** axioms and rules for all operators $[R]$ $(R \in \mathcal{R})$ and adding the following

(Comp)	$\vdash [\alpha\, ;\, \beta]A \leftrightarrow [\alpha][\beta]A$
(Alt)	$\vdash [\alpha \cup \beta]A \leftrightarrow [\alpha]A \wedge [\beta]A$
(Mix)	$\vdash [\alpha^*]A \to A \wedge [\alpha][\alpha^*]A$
(Ind)	$\vdash [\alpha^*](A \to [\alpha]A) \to (A \to [\alpha^*]A)$
(Test)	$\vdash [A?]B \leftrightarrow (A \to B)$

(Mix) and (Ind) are sometimes referred to as the *Segerberg axioms*; together they capture the fact that the program α^* is α iterated finitely often.

Extensions of **K** are obtained by adding further axioms and/or rules to **K**. For instance, the logic **S4** is obtained by adding the following axioms to **K**:

(T) $\quad \vdash \Box A \to A$

(4) $\quad \vdash \Box A \to \Box\Box A$

2.8. Definition *(Classical Modal Systems).* The so-called *classical* modal system [Chellas 1980] is a restriction of **K** in which the K axiom (2.2) and the necessitation rules (2.4) are replaced by:

$$\text{(RE)}\ \frac{\vdash A \leftrightarrow B}{\vdash \Box A \leftrightarrow \Box B} \tag{2.5}$$

This rule ensures that the box operator is syntax independent in the sense that it does not distinguish equivalent formulae.

In a similar way as with extensions of the modal logic **K**, one can build logics as extensions of the system characterized by **RE**. Examples of other rules are:

$$\text{(RM)}\ \frac{\vdash A \to B}{\vdash \Box A \to \Box B} \tag{2.6}$$

$$\text{(RR)}\ \frac{\vdash (A \wedge B) \to C}{\vdash (\Box A \wedge \Box B) \to \Box C} \tag{2.7}$$

2.9. Definition *(Axioms for Temporal Logic).* The smallest *basic temporal logic* $\mathbf{K}_t$ consists of the **K**-axioms for each of $[F]$ and $[P]$, as well as the 4 axiom for both $[F]$ and $[P]$, and the following two axioms that are meant to express that the temporal operators explore the same flow of time, but in opposite directions.

(Conv1) $\quad \vdash A \to [P]\langle F\rangle A$

(Conv2) $\quad \vdash A \to [F]\langle P\rangle A$

The axioms for linear time temporal logic, **LTL**, consist of the **K** axioms for $\Box$, $\bigcirc$ and $\bullet$, the Segerberg axioms for $\Box$ and $\bigcirc$, and for $\blacksquare$ and $\bullet$, as well as axioms for functionality and U and S; see [Goldblatt 1992] or [Clarke and Schlingloff 2001] (Chapter 24 of this Handbook) for details.

2.10. Definition *(Axioms for Intuitionistic Logic).* An axiom system **IL** for intuitionistic logic is given by

$\vdash A \to (B \to A)$

$\vdash (A \to B) \to ((A \to (B \to C)) \to (A \to C))$

$\vdash A \to (B \to A \wedge B)$

$\vdash (A \wedge B) \to A$

$\vdash (A \wedge B) \to B$

$\vdash A \to (A \vee B)$

$\vdash B \to (A \vee B)$

$\vdash (A \to C) \to ((B \to C) \to (A \vee B \to C)$

$\vdash (A \to B) \to ((A \to \neg B) \to \neg A)$

$\vdash (A \to (\neg A \to B)$.

The only deduction rule is modus ponens (2.3).

2.2. Semantics

We will first give a general definition, and then instantiate it for particular logics.

Many two-valued nonclassical logics have a characterization in terms of a so-called *possible worlds semantics.*

2.11. DEFINITION *(Possible World Semantics).* The basis for a possible world semantics is a *frame* $\mathcal{F} = (W, S)$ consisting of a non-empty set W of worlds or states and some structure S on these worlds. S may be just a binary relation between worlds (the *accessibility relation*) as in normal modal logics and in intuitionistic logic, or a binary relation satisfying special properties (such as the later-than relation in temporal logic), or a set of binary relations as in multi-modal logic, or a relation between worlds and sets of worlds (the *neighborhood relation*) as in classical modal logic, or a ternary relation between worlds as in relevance logic.

An *interpretation* (or *model*) $\Im$ based on a frame $\mathcal{F}$ associates a classical interpretation with each world, sometimes subject to some restrictions. In addition, an interpretation may contain an *actual world* and, if the logic has quantifiers, a variable assignment.

Let us now instantiate the general definition for some specific logics. We start with (uni-) modal logic.

2.12. DEFINITION *(Semantics of Modal Logic).* A frame, or *Kripke frame*, for the (uni-) modal language with just $\Diamond$ and $\Box$ is a structure $\mathcal{F} = (W, R)$ consisting of a non-empty set of worlds W and a binary accessibility relation R on W.

An interpretation, or *Kripke model*, $\Im = (\mathcal{F}, P)$ consists of a frame $\mathcal{F}$, and a *valuation* or *predicate assignment* P which assigns to each proposition letter p the subset of worlds where p is interpreted as *true*. Each model induces a *satisfiability relation* $\models$ that determines the truth of a formula at a given *actual world* $w \in W$.

The *satisfiability relation* for the basic uni-modal language is given by the following clauses.

$$\begin{array}{rll}
\Im, w \models p & \text{iff} & w \in P(p) \quad \text{in case } p \text{ is a predicate symbol} \\
\Im, w \models \neg A & \text{iff} & \text{not } \Im, w \models A \\
\Im, w \models A \wedge B & \text{iff} & \Im, w \models A \ \& \ \Im, w \models B \\
\Im, w \models A \vee B & \text{iff} & \Im, w \models A \text{ or } \Im, w \models B \\
\Im, w \models \Box A & \text{iff} & \forall v\,(R(w,v) \Rightarrow \Im, v \models A) \\
\Im, w \models \Diamond A & \text{iff} & \exists v\,(R(w,v) \ \& \ \Im, v \models A).
\end{array}$$

A *frame* for a multi-modal language whose collection of modal operators is $\{\langle\alpha\rangle \mid \alpha \in \mathcal{R}\}$ is a tuple $\mathcal{F} = (W, \{R_\alpha \mid \alpha \in \mathcal{R}\})$. The satisfaction relation for multi-modal languages is defined just as for the uni-modal language, albeit that every modal operator $\langle\alpha\rangle$ is interpreted using its own relation R_α.

A *model* for the language of propositional dynamic logic is a structure $\Im = (W, \{R_\alpha \mid \alpha \in \mathcal{R}\}, P)$ just as for multi-modal logic, but the difference is that the relations R_α (for α non-atomic) are required to satisfy certain structural constraints:

$$\begin{array}{rcl}
R_{\alpha;\beta} & = & R_\alpha \circ R_\beta, \text{ the relational composition of } R_\alpha \text{ and } R_\beta \\
R_{\alpha\cup\beta} & = & R_\alpha \cup R_\beta \\
R_{\alpha^*} & = & \bigcup_n (R_\alpha)^n, \text{ where } (R_\alpha)^0 \text{ is the identity relation, and} \\
 & & (R_\alpha)^{n+1} = (R_\alpha)^n \circ R_\alpha \\
R_{A?} & = & \{(w,w) \mid \Im, w \models A\}
\end{array}$$

A formula ϕ is a *semantic consequence* of a particular logic, characterized by a particular frame class, if and only if it is *true* for all such interpretations, i.e., for all frames in a given class, for all actual worlds and for all predicate assignments.

Particular logics are characterized by certain frame classes, or in more complex cases, by certain model classes. To be *characterized by a frame class* means that the set of semantic consequences of this logic consists of all formulae which are interpreted as *true* in all interpretations based on all frames of this class. For example, **K** is characterized by the class of frames (W, R) where R is an arbitrary binary relation between worlds. Further examples for axioms and the corresponding properties of the accessibility relation are given in Table 2. It is not always possible to characterize a logic just by a frame class [van Benthem 1983]. More complex logics may require special conditions on interpretations, too.

name	axiom	property of R
4	$\Box A \to \Box\Box A$	transitivity
5	$\Diamond A \to \Box\Diamond A$	euclidean[1]
T	$\Box A \to A$	reflexivity
B	$A \to \Box\Diamond A$	symmetry
D	$\Box A \to \Diamond A$	seriality

Table 2: Axioms and their frame properties

2.13. DEFINITION *(Semantics of Temporal Languages).* The semantics of the temporal languages considered in this chapter are defined as follows. A *temporal frame* is a structure $\mathcal{F} = (W, \leq)$ where W should now be thought of as a collection of points in time, and $\leq$ is a transitive binary relation on W representing the 'later-than' relation.

The key aspect of the truth definition is in the clauses for the modal operators:

$$\begin{array}{lcl}
\Im, w \models \langle F\rangle A & \text{iff} & \exists v\,(w \leq v \,\&\, \Im, v \models A) \\
\Im, w \models \langle P\rangle A & \text{iff} & \exists v\,(v \leq w \,\&\, \Im, v \models A).
\end{array}$$

[1]A relation R is called *euclidean* if it satisfies $\forall v, w, u\,(R(v,w) \wedge R(v,u) \to R(w,u))$.

Thus, $\langle F\rangle$ and $\langle P\rangle$ are interpreted using the same relation $\leq$.

Models for linear time temporal logic are structures $\Im = (W, <, P)$ where $(W, <)$ is an initial segment of the natural numbers $(\omega, <)$ with its natural ordering, and P is a valuation. The truth conditions for the temporal operators are as follows:

$$\begin{aligned}
\Im, w \models \bigcirc A &\quad \text{iff} \quad \Im, w+1 \models A \\
\Im, w \models \Box A &\quad \text{iff} \quad \forall v\,(w < v \Rightarrow \Im, v \models A) \\
\Im, w \models U(A,B) &\quad \text{iff} \quad \exists v\,(w \leq v \,\&\, \Im, v \models A \,\&\, \forall u\,(w \leq u < v \Rightarrow \Im, u \models B)) \\
\Im, w \models \bullet A &\quad \text{iff} \quad w > 0 \,\&\, \Im, w-1 \models A,
\end{aligned}$$

while $\blacksquare$ and S are simply backward looking versions of $\Box$ and U.

2.14. DEFINITION *(Semantics of Intuitionistic and Relevance Logic).* The semantics of intuitionistic logic can also be defined in terms of possible worlds, but we need to impose a restriction on assignments. An *intuitionistic frame* $\mathcal{F} = (W, R)$ consists of a non-empty set of worlds, together with a *reflexive* and *transitive* relation R. An interpretation $\Im = (\mathcal{F}, P)$ is similar to an interpretation for modal logic, but whenever P assigns *true* to some predicate symbol p at some world w then it must assign *true* to p at all w-accessible worlds.

The possible worlds semantics of relevance logic is more complex. It has a *ternary* accessibility relation R, a unary function $(\cdot)^*$ and a distinguished world 0. A *relevant frame* is therefore a 4-tuple $\mathcal{F} = (W, R, (\cdot)^*, 0)$ consisting of a non-empty set of worlds, a distinguished world $0 \in W$, a ternary accessibility relation R, and a unary function $(\cdot)^*$ satisfying the following conditions:

$$\begin{aligned}
&\forall x\, R(0, x, x) \\
&\forall x, y, z\,(R(x,y,z) \rightarrow R(y,x,z)) \\
&\forall x, y, z, u\,(\exists v\,(R(x,y,v) \wedge R(v,z,u)) \rightarrow \exists v\,(R(x,v,u) \wedge R(y,z,v))) \\
&\forall x\, R(x, x, x) \\
&\forall x, y, z, x'\,(R(x,y,z) \wedge R(0,x',x) \rightarrow R(x',y,z)) \\
&\forall x\, x^{**} = x \\
&\forall x, y, z\,(R(x,y,z) \rightarrow R(x, z^*, y^*)).
\end{aligned}$$

An *interpretation* consists of a frame and a predicate assignment P with a restriction which is similar to the restriction for intuitionistic logic: whenever P assigns *true* to some predicate symbol p at some world w then it must assign *true* to p at all worlds u with $R(0, w, u)$. The non-standard parts of the satisfiability relation are

$$\begin{aligned}
\Im, w \models A \rightarrow B &\quad \text{iff} \quad \forall u, v\, R(w,u,v) \Rightarrow \Im, u \models A \Rightarrow \Im, v \models B, \text{ and} \\
\Im, w \models \neg A &\quad \text{iff} \quad \Im, w^* \not\models A
\end{aligned}$$

2.15. DEFINITION *(Semantics of Classical Modal Logic).* The axiom K is valid in all modal logics whose semantics are based on binary accessibility relations. If this axiom is undesirable for some reason, a different kind of semantics is necessary, *neighborhood semantics*, also called *minimal model semantics* [Chellas 1980]. In its weakest version, it just ensures that the modal operators cannot distinguish between equivalent formulae, or, in other words, if $A \leftrightarrow B$ is valid then $\Box A \leftrightarrow \Box B$ is valid, too.

A neighborhood frame $\mathcal{F} = (W, N)$ consists of a non-empty set of possible worlds and a *neighborhood relation* N between worlds and *sets of worlds*. The interpretation for the classical connectives is the same as for normal modal logics. Only the interpretation of the modal operators changes. For the weaker version of classical modal logics the interpretation is:

$$\begin{array}{lll} \Im, w \models \Box A & \text{iff} & \exists V\,(N(w,V)\ \&\ \forall v\,(v \in V \Leftrightarrow \Im, v \models A)) \\ \Im, w \models \Diamond A & \text{iff} & \forall V\,(N(w,V) \Rightarrow \exists v\,(v \in V \Leftrightarrow \Im, v \models A)). \end{array}$$

If the rule **RM** (2.6)

$$\frac{\vdash A \to B}{\vdash \Box A \to \Box B}$$

holds, then the following stronger semantics is to be used instead

$$\begin{array}{lll} \Im, w \models \Box A & \text{iff} & \exists V\,(N(w,V)\ \&\ \forall v\,(v \in V \Rightarrow \Im, v \models A)) \\ \Im, w \models \Diamond A & \text{iff} & \forall V\, N((w,V) \Rightarrow \exists v\,(v \in V\ \&\ \Im, v \models A)). \end{array}$$

The weak version of the minimal model semantics states that $\Box A$ is true in a world w iff the truth set of A is one of w's neighborhoods. Therefore, equivalent formulae are not distinguishable by the box operator. The stronger version of the semantics weakens this requirement: the truth set of A only needs to be a superset of one of w's neighborhoods.

2.16. Definition *(Quantified Modal Logic).* The semantics of quantified modal logic has the same kind of frames $\mathcal{F} = (W, R)$ with a binary accessibility relation as normal propositional modal logic. An interpretation $\Im = (\mathcal{F}, P, V)$ consists of a frame $\mathcal{F}$, an assignment P associating with each world w a classical predicate logic interpretation $P(w)$, and a variable assignment V.

Each predicate logic interpretation $P(w)$ may associate a different meaning to the function and predicate symbols. *Rigid* symbols are function and predicate symbols whose interpretation does not change from world to world, otherwise they are called *flexible* symbols. $P(w)$ may even associate a different domain with each world (varying domain interpretation). Alternatively, the domain may remain the same for all worlds (constant domain) or increase (decrease) when moving from a world to an accessible world.

The interpretation $P(w)(p)$ of a function or predicate symbol p must be defined for all elements of all domains, not only for the domain associated with the world w, otherwise the following problem arises. For the formula $\exists x\, \Diamond p(x)$ the interpretation of the existential quantifier chooses an element a of the domain of the actual world

w as the assignment for the variable x. The $\Diamond$-operator guides us to another world v. We can only interpret $p(x)$ in v's domain if $P(v)(p)$ is also defined for objects in the other world's domains.

Let $\mathcal{D}(w)$ denote w's domain, let $\Im[x/a]$ be just like $\Im$, except that the variable x is mapped to a, and let $\Im(t, w)$ represent the interpretation of the term t in the world w. Then the satisfiability relation $\models$ is defined as

$$\begin{array}{rll} \Im, w \models q(t_1, \dots, t_n) & \text{iff} & (\Im(t_1, w), \dots, \Im(t_n, w)) \in P(q) \\ & & \qquad \text{in case } q \text{ is a predicate symbol} \\ \Im, w \models \forall x\, A & \text{iff} & \forall a\, (a \in \mathcal{D}(w) \Rightarrow \Im[x/a], w \models A) \\ \Im, w \models \exists x\, A & \text{iff} & \exists a\, (a \in \mathcal{D}(w)\ \&\ \Im[x/a], w \models A). \end{array}$$

After this short summary of the semantics of the most prominent modal-like logics, we come to the core of this chapter: how to encode these logics into predicate logic.

3. Encoding consequence relations

If we want to encode a given source logic into a target logic, and if the source logic is given as a Hilbert system, we can try to use the target logic, $\mathbf{L}_2$, as a metalevel language for the source logic, $\mathbf{L}_1$. The well-formed formulae (wffs) of $\mathbf{L}_1$ become terms of $\mathbf{L}_2$ and the axioms and theorems of $\mathbf{L}_1$ become theorems of $\mathbf{L}_2$. Let us make this more precise.

A Hilbert system HS — with all predicate symbols implicitly universally quantified, and where the inference rules have only finitely many (positive) input formulae and no side conditions — can be encoded in first-order predicate logic (FO) in the following manner.

3.1. Example. Take the source logic to be the uni-modal logic $\mathbf{K}$, and the target logic is simply FO. To cater for the operators present in the source logic, we take function symbols $f_\wedge$, $f_\vee$, $f_\rightarrow$, $f_\neg$, $f_\Box$, $f_\Diamond$ in the vocabulary of the target logic. We form all terms out of constants $\{c_1, c_2, c_3, \ldots\}$ and variables $\{p_1, p_2, p_3, \ldots\}$. We define a translation $(\cdot)^*$ assigning to each modal wff A a FO-term A^*, as detailed in Table 3.

We can identify the axioms and theorems of the source logic $\mathbf{L}_1$ by defining a predicate $\mathit{Th}(x)$ (x is a *theorem*) on the terms obtained as translations of $\mathbf{L}_1$.More precisely, in the syntactic way of encoding a logic, formulae are represented as FO-terms; HS-predicate symbols become universally quantified FO-variables, while the HS-connectives become FO-function symbols, and the consequence relation $\vdash$ becomes a unary FO-predicate Th (for theoremhood). The axioms are encoded as unit clauses and the inference rules are encoded as Horn-formulae.

3.2. Example. We build on Example 3.1 to encode a Hilbert system for the modal logic $\mathbf{S4}$. Recall that $\mathbf{S4}$ can be axiomatized by adding the following axioms on top of the axioms and rules given for $\mathbf{K}$ in Definition 2.7:

$$
\begin{aligned}
(c_n)^* &= a_{c_n}, \text{ for the } n\text{th constant } c_n \\
(p_n)^* &= x_{p_n}, \text{ for the } n\text{th variable } p_n \\
(A \wedge B)^* &= f_\wedge(A^*, B^*) \\
(A \to B)^* &= f_\to(A^*, B^*) \\
(A \vee B)^* &= f_\vee(A^*, B^*) \\
(\neg A)^* &= f_\neg(A^*) \\
(\Box A)^* &= f_\Box(A^*) \\
(\Diamond A)^* &= f_\Diamond(A^*).
\end{aligned}
$$

Table 3: Syntactic encoding

(T) $\vdash \Box A \to A$
(4) $\vdash \Box A \to \Box\Box A$
Take the metalevel translation as specified in Example 3.1. We can identify the axioms and theorems of **S4** by defining a predicate $Th(x)$ (x is a *theorem*) in classical logic on the terms (wff) of $\mathbf{S4}^*$ (the translation of **S4**).

As the theory τ_* of classical logic we take

$$\text{(universal closure of) } Th(t^*)$$

where t^* is a variable translation of any **S4** axiom, e.g., $t^* = f_\to(x, f_\to(y, x))$ as translation of $A \to (B \to A)$, plus the metalevel translation of modus ponens and the necessitation rule:

$$\forall x, y\ (Th(x) \wedge Th(f_\to(x, y)) \to Th(y)) \tag{3.1}$$

$$\forall x\ (Th(x) \to Th(f_\Box(x))). \tag{3.2}$$

Then it follows, for all modal formulae A, that

$$\mathbf{S4} \vdash A \quad \text{iff} \quad \tau_* \vdash Th(A^*).$$

How does a FO theorem prover behave when fed with the syntactic encoding of a Hilbert system? Observe first that a UR-resolution or a hyperresolution sequence with the FO encoding of a Hilbert system corresponds directly to a derivation in the system itself [McCune and Wos 1992]. Since the clause set obtained from an encoded Hilbert system is a Horn clause set, the (unique) Herbrand model represents the theorems of the Hilbert system. Now, the syntactic way of encoding a logic does not automatically provide a decision procedure for theoremhood within the source logic, even if this problem is decidable in principle. In particular, UR-resolution with a syntactically encoded Hilbert system just enumerates the — usually infinitely many — theorems of the logic. A non-theorem can therefore never be refuted this way.

Experiments with first-order theorem provers have shown that the search space for this kind of encoding is very large [McCune and Wos 1992], and therefore the theorem provers are very slow.

Ohlbach [1998*b*] has investigated a transformation which can eliminate self-resolving clauses like the condensed detachment clause (3.1) from a clause set in a satisfiability preserving way. Self-resolving clauses can be deleted from a clause set if sufficiently many other clauses, which are consequences of the self-resolving clauses and the other clauses, are added. For the transitivity clause, for example, it is sufficient to add for each positive literal in the other clauses one single resolvent with a selected negative literal of the transitivity clause. Afterwards, transitivity is not needed anymore. This transformation can also be applied to the condensed detachment clause.

Unfortunately, it turns out that infinitely many resolvents are to be added before one can delete a condensed detachment clause. The clauses to add are (for each positive literal) a resolvent with the first literal in each of the infinitely many clauses below, which are themselves self-resolvents between the first and third literal of condensed detachment.

$$\begin{array}{l}
\neg\, Th(f_{\rightarrow}(x,y)), \neg\, Th(x),\, Th(y)\\
\neg\, Th(f_{\rightarrow}(x, f_{\rightarrow}(z_1,z_2))), \neg\, Th(x), \neg\, Th(z_1),\, Th(z_2)\\
\ldots\\
\neg\, Th(f_{\rightarrow}(x, f_{\rightarrow}(z_1, f_{\rightarrow}(\ldots z_j)))), \neg\, Th(x) \neg\, Th(z_1), \ldots, \neg\, Th(z_{j-1}),\, Th(z_j)\\
\ldots
\end{array}$$

3.3. EXAMPLE. We illustrate this transformation with Łukasiewicz's single axiom axiomatization of the implicational fragment of propositional logic [Łukasiewicz 1970, p. 295]:

$$\vdash ((x \rightarrow y) \rightarrow z) \rightarrow ((z \rightarrow x) \rightarrow (u \rightarrow x)). \tag{3.3}$$

The transformation yields

$$\begin{array}{l}
Th(f_{\rightarrow}(f_{\rightarrow}(f_{\rightarrow}(x,y),z), f_{\rightarrow}(f_{\rightarrow}(z,x), f_{\rightarrow}(u,x))))\\
Th(f_{\rightarrow}(f_{\rightarrow}(x,y),z) \rightarrow Th(f_{\rightarrow}(f_{\rightarrow}(z,x), f_{\rightarrow}(u,x)))\\
Th(f_{\rightarrow}(f_{\rightarrow}(x,y),z) \wedge Th(f_{\rightarrow}(z,x)) \rightarrow Th(f_{\rightarrow}(u,x))\\
Th(f_{\rightarrow}(f_{\rightarrow}(x,y),z) \wedge Th(f_{\rightarrow}(z,x)) \wedge Th(u) \rightarrow Th(x)\\
Th(f_{\rightarrow}(f_{\rightarrow}(f_{\rightarrow}(u_1,u_2),y),z) \wedge Th(f_{\rightarrow}(z, f_{\rightarrow}(u_1,u_2))) \wedge Th(u) \wedge\\
\qquad Th(u_1) \rightarrow Th(u_2)\\
Th(f_{\rightarrow}(f_{\rightarrow}(f_{\rightarrow}(u_1, f_{\rightarrow}(u_2,u_3)),y),z) \wedge Th(f_{\rightarrow}(z, f_{\rightarrow}(u_1, f_{\rightarrow}(u_2,u_3)))) \wedge\\
\qquad Th(u) \wedge Th(u_1) \wedge Th(u_2) \rightarrow Th(u_3)\\
\ldots,
\end{array}$$

which is infinite, but with a very regular structure. With some standard and well-known tricks one can convert it into a finite clause set suitable as input to any resolution-based theorem prover. Longer subformulae are replaced with newly defined predicates ($q_1, \ldots, q_4$) and the recursive structure of the infinitely many formulae above is turned into a 'generator clause' (the third but last clause below). The result is

$$\begin{array}{l}
Th(f_\to(f_\to(f_\to(x,y),z),f_\to(f_\to(z,x),f_\to(u,x)))) \\
Th(f_\to(f_\to(x,y),z) \to q_1(x,z) \\
q_1(x,z) \to Th(f_\to(f_\to(z,x),f_\to(u,x))) \\
q_1(x,z) \wedge Th(f_\to(z,x)) \to q_2(x) \\
q_2(x) \to Th(x) \\
q_2(x) \to q_3(x) \\
q_3(f_\to(u,v)) \wedge Th(u) \to q_4(v) \\
q_4(v) \to q_3(v) \\
q_4(v) \to Th(v).
\end{array}$$

Experiments have shown that proving theorems from this clause set instead of the original condensed detachment clause can speed up the theorem prover by a factor of 50 [Ohlbach 1998*b*]. But there is no guarantee. In other cases this transformation has slowed theorem provers down considerably.

To sum up, the direct syntactic encoding of Hilbert systems into FO is certainly not the optimal way of proving theorems in these logics. Nevertheless, for experimenting with new logics or for getting alternative formulations of a given logic, this encoding together with a fast FO theorem prover provides a very useful tool.

One possible obstacle for such experiments is that there is no general recipe for encoding side conditions of inference rules in FO. This is usually quite complicated because the side conditions may refer to the syntactic structure of a formula. Here is a particular example.

3.4. EXAMPLE. In modal logic one of the best-known inference rules with a side condition is Gabbay's *irreflexivity rule* [Gabbay 1981*b*]

$$\frac{\vdash \neg(\Box p \to p) \to A}{\vdash A} \quad \text{if } p \notin A.$$

To encode the irreflexivity rule, we need a predicate $in(p, A)$ which checks whether p occurs in A. The FO-axiom for in would look like

$$\begin{array}{rcl}
in(p,A) & \leftrightarrow & p = A \ \vee \\
 & & (\exists B\ A = f_\neg(B) \wedge in(p,B)) \ \vee \\
 & & (\exists A_1, A_2\ A = f_\to(A_1,A_2) \wedge (in(p,A_1) \vee in(p,A_2))) \ \vee \ \ldots
\end{array}$$

Adding this axiom, however, requires equational reasoning. Most FO theorem provers have serious difficulty with this formulation.

Binary consequence relations $\phi \vdash \psi$, where both ϕ and ψ are single formulae, can be encoded in a similar way as the unary consequence relation of Hilbert systems. Instead of the unary predicate $Th(x)$ ('x is a theorem'), we use the binary predicate $D(x, y)$ (y can be derived from x).

If ϕ and ψ are more complex datastructures, one has to axiomatize these datastructures. For instance, if the datastructure is a list (as in linear logic), a constant *empty* needs to be introduced as well as an associative function symbol *append* that takes the FO-encoded formulae as arguments. The axiom $X, A \vdash A$, for example, would be encoded as

$$\forall X, A\, D(\mathit{append}(X, A), \mathit{append}(A, \mathit{empty})).$$

Unfortunately, automated reasoning based on these kinds of axioms, and either paramodulation or theory unification for the associativity property of *append*, becomes much more complicated and inefficient than in the Hilbert system case. Therefore, we think that it is not worthwhile to go into further details here.

The situation is even worse if some of the input consequences of a consequence relation are negated. *not* $\phi \vdash \psi$ means that the input condition is satisfied if $\phi \vdash \psi$ is *not* derivable. This cannot be encoded using the FO-negation, because it requires a proof that $\phi \vdash \psi$ is not a theorem. Another metalevel, which encodes the FO-derivability relation itself as a predicate, or some Prolog-like operational treatment (negation by failure) may help here.

To conclude, then, the syntactic encoding of logics is an easy and flexible way of encoding a nonclassical logic in FO, which imposes only a few conditions on the source logics being encoded. This flexibility proves to be an advantage as well as a disadvantage: only very little information about the source logic is actually encoded into the target logic by means of syntactic encodings.

4. The standard relational translation

If a given nonclassical logic comes with a possible worlds semantics of the kind described in Section 2.2, there is an alternative to the syntactic encoding that we have seen in Section 3: we can encode the semantics of the source logic by simply transcribing it inside the target logic.

Below we will introduce the idea using the uni-modal language, as well as a number of other ones. In most cases the semantics-based relational encoding is fairly simple and the reader should have no difficulties in doing something similar for other logics. After the examples, we will explore the implications of having this translation for understanding the expressive power of the source logics being encoded, and for the purposes of reasoning with the source logic. The latter will also form the motivation for refinements of the relational translation.

4.1. The basic case

The prototypical example for the standard or relational translation is propositional uni-modal logic. Recall the truth conditions for the modal operators:

$$\begin{aligned} \Im, w \models \Box A \quad &\text{iff} \quad \forall v\, (R(w,v) \Rightarrow \Im, v \models A) \\ \Im, w \models \Diamond A \quad &\text{iff} \quad \exists v\, (R(w,v)\ \&\ \Im, v \models A). \end{aligned}$$

From these definitions we can derive the definition of the *standard* or *relational translation* $ST_m(w, A)$.

4.1. Definition *(Standard Relational Translation).* The source logic is uni-modal logic, and the target logic is FO; its vocabulary of the target logic consists of a binary predicate symbol R to represent the accessibility relation, and unary predicate symbols to represent proposition letters. Then, the recursive definition for the *standard relational translation* is

$$\begin{aligned}
ST_m(w, p) &= p(w) \\
ST_m(w, \neg A) &= \neg ST_m(w, A) \\
ST_m(w, A \wedge B) &= ST_m(w, A) \wedge ST_m(w, B) \\
ST_m(w, A \vee B) &= ST_m(w, A) \vee ST_m(w, B) \\
ST_m(w, \Box A) &= \forall v\, (R(w, v) \to ST_m(v, A)) && (4.1) \\
ST_m(w, \Diamond A) &= \exists v\, (R(w, v) \wedge ST_m(v, A)). && (4.2)
\end{aligned}$$

Now, on the semantic side, Kripke models have all the ingredients to serve as models for the first-order language that we have just defined: they have a binary relation for the binary predicate symbol, and they have valuations to cater for the unary predicate symbols.

4.2. Theorem. *Let A be a modal formula. Then the following hold:*

1. *For any model $\Im$ and state w we have that $\Im, w \models A$ iff $\Im \models ST_m(w, A)$, where $\models$ on the left-hand side is the modal satisfiability relation, and $\models$ on the right-hand side denotes first-order satisfiability.*
2. *$\models A$ iff $\models \forall x\, ST(x, A)$*

Item 1 of the theorem says that when we interpret modal formulae on models we can view them as first-order formulae, while item 2 says that a modal formula is valid on all Kripke models iff the universal closure of its standard translation is valid on all FO models.

4.3. Example. $ST_m(w, A)$ yields a first-order formula and can be submitted to a first-order theorem prover. As an example, in the modal logic $\mathbf{K}$, the axiom $\Box(p \to q) \to (\Box p \to \Box q)$ is a theorem. Its standard translation $\forall w\ (\forall v\ R(w, v) \to (p(v) \to q(v)) \to ((\forall v\ R(w, v) \to p(v)) \to (\forall v\ R(w, v) \to q(v)))$ is therefore a FO theorem.

While the standard relational translation was implicit in much of the work in modal and temporal logic going on in the early 1970s, the first systematic study seems to be due to [van Benthem 1976].

Recall from Table 2 that many of the well-known modal logics are characterized by first-order definable frame classes. For such logics Theorem 4.2 may be specialized as follows.

4.4. THEOREM. *Let A be a uni-modal formula, and let $B_1, \ldots, B_n$ be characterized by first-order frame conditions $\tau_{B_1}, \ldots, \tau_{B_n}$, respectively. Then A is a semantic consequence of $\mathbf{KB}_1 \ldots \mathbf{B}_n$ iff $(\tau_{B_1} \wedge \cdots \wedge \tau_{B_n}) \rightarrow \forall x\, ST_m(x, A)$ is a theorem of first-order logic.*

4.5. EXAMPLE. **S4** is complete for several semantic interpretations. The one that we are interested in here is the usual Kripke semantics, where R is a reflexive and transitive accessibility relation, and $w \vDash \Box A$ iff $\forall u\,(wRu \Rightarrow u \vDash A)$. Let τ be the FO theory

$$\tau = \{\forall x\, R(x,x), \forall x, y, z\, (R(x,y) \wedge R(y,z) \rightarrow R(x,z))\}.$$

Then we have, for every modal formula A,

$$\mathbf{S4} \models A \text{ iff } \tau \models \forall x\, ST_m(x, A).$$

Before addressing the issue of reasoning with (translated) modal formulae in greater depth, we will look at uses of the standard relational translation for nonclassical logics other than uni-modal logic.

4.2. Further examples

The ideas underlying the standard translation are not restricted to the basic modal logic **K**. Below we list some examples.

4.6. EXAMPLE *(Multi-Modal Logic).* Definition 4.1 can easily be extended to multi-modal logic; we simply make sure that the vocabulary of the target logic has 'enough' binary relation symbols R_a, R_b, …: one for every diamond $\langle a \rangle$ and box $[a]$. Then, the key clause in the translation becomes

$$ST_m(w, [a]A) = \forall v\, (R_a(w,v) \rightarrow ST_m(v, A)).$$

So, the formula $\langle a \rangle [b] \langle a \rangle p$ is translated as

$$\exists v\, (R_a(w,v) \wedge \forall u\, (R_b(v,u) \rightarrow \exists t\, (R_a(u,t) \wedge (p(t))))).$$

We leave it to the reader to establish equivalences analogous to those in Theorem 4.2.

4.7. EXAMPLE *(Propositional Dynamic Logic).* If we want to provide a relational translation for propositional dynamic logic (PDL), we have to change our target logic because of the presence of programs of the form α^*. Recall that a program α^* is interpreted using the reflexive, transitive closure of R_α. But the reflexive, transitive closure of a relation is not a first-order definable relation (see Definition 2.12).

What is a suitable target logic here? There are many options, but to motivate our choice recall the definition of the meaning of a PDL program α^*:

$$R_{\alpha^*} = \bigcup_n (R_\alpha)^n,$$

where R^n_α is defined by $R^0(x,y)$ iff $x = y$, and $R^{n+1}(x,y)$ iff $\exists z\,(R^n(x,z) \wedge R(z,y))$. If we are allowed to write infinitely long disjunctions, we see how to capture the meaning of an iterated program α^*:

$$(R_\alpha)^* xy \text{ iff } (x = y) \ \vee R_\alpha xy \vee \bigvee \exists z_1 \ldots z_n\,(R_\alpha x z_1 \wedge \cdots \wedge R_\alpha z_n y).$$

The infinitary predicate language allows us to write down such formulae. More precisely, in $\mathcal{L}_{\omega_1\omega}$ one is allowed to form formulae just as in first-order logic, but in addition countably infinite disjunctions and conjunctions are allowed.

Finally, then, the relational translation $ST_p(\cdot,\cdot)$ of PDL into $\mathcal{L}_{\omega_1\omega}$ has the following modal clauses:

$$ST_p(w, \langle\alpha\rangle A) = \exists v\,(ST_p(wv,\alpha) \wedge ST_p(v,A)),$$

where the translation $ST_p(wv,\alpha)$ of programs α requires two free variables:

$$\begin{aligned}
ST_p(wv,a) &= R_a(w,v) \text{ (and similarly for other pairs of variables)}\\
ST_p(wv,\alpha \cup \beta) &= ST_p(wv,\alpha) \vee ST_p(wv,\beta)\\
ST_p(wv,\alpha\,;\beta) &= \exists u\,(ST_p(wu,\alpha) \wedge ST_p(uv,\beta))\\
ST_p(wv,\alpha^*) &= (w = v) \vee ST_p(wv,\alpha) \vee\\
&\quad \bigvee_{n\geq 1} \exists u_1 \ldots u_n\,(ST_p(wu_1,\alpha) \wedge \cdots \wedge ST_p(u_n v,\alpha)).
\end{aligned}$$

It may be proved that every PDL-formula ϕ is equivalent to its relational translation on the class of PDL-models.

4.8. EXAMPLE *(Linear Temporal Logic).* For (linear) temporal logic, too, a relational translation may be obtained by simply transcribing its truth definitions in a suitable target logic. Here we can simply take FO for the target logic, and its vocabulary should contain a binary relation symbol $\leq$ as well as the usual unary predicate symbols. Then, the relational translation simply becomes

$$\begin{aligned}
ST_t(w, U(A,B)) &= \exists v\,(w \leq v \wedge ST_t(v,A) \wedge\\
&\qquad \forall u\,(w \leq u \wedge u < v \rightarrow ST_t(u,B))).
\end{aligned}$$

Observe that this translation needs 2 quantifiers (and three variables), whereas the translation for uni-modal logic only needs 1 quantifier (and 2 variables). We will come back to this issue in Subsection 4.3.

4.9. EXAMPLE *(Intuitionistic Logic).* Recall from Definition 2.14 that the semantics of intuitionistic logic is based on a reflexive, transitive relation and that assignments need to satisfy the following restriction

$$\forall w,u\,(p(w) \wedge R(w,u) \rightarrow p(u)) \tag{4.3}$$

for each translated proposition letter p. The relational translation for all connectives except for the implication are as in the modal case. The intuitionistic semantics of the implication

$$\begin{aligned} \Im, w \models A \rightarrow B \quad &\text{iff} \quad \forall v\,(R(w,v) \Rightarrow \Im, v \models A \Rightarrow (\Im, v \models B)) \\ \Im, w \models \neg A \quad &\text{iff} \quad \forall v\,(R(w,v) \Rightarrow \Im, v \not\models A) \end{aligned}$$

yields the relational translation

$$ST_i(w, A \rightarrow B) = \forall v\,(R(w,v) \rightarrow (ST_i(v,A) \rightarrow ST_i(v,B))) \tag{4.4}$$

$$ST_i(w, \neg A) = \forall v\,(R(w,v) \rightarrow \neg ST_i(v,A)). \tag{4.5}$$

4.10. Example *(Relevance Logic).* The semantics of relevance logic was given in Definition 2.14. From the definition, the relational translation can be derived:

$$\begin{aligned} ST_r(w, A \rightarrow B) &= \forall u, v\,(R(w,u,v) \rightarrow (ST_r(u,A) \rightarrow ST_r(v,B))) \\ ST_r(w, \neg A) &= \neg ST_r(w^*, A). \end{aligned}$$

A formula A is a theorem of relevance logic if it is valid in the distinguished world 0. To check this with a FO theorem prover, $ST_r(0, A)$ has to be proved from the axioms about R and $(\cdot)^*$ that were listed in Definition 2.14. Further, A *entails* B in relevance logic if and only if for all interpretations and worlds w, if $\Im, w \models A$ then $\Im, w \models B$ holds. This can be checked by using a FO-theorem prover to derive $\forall w\,(ST_r(w,A) \rightarrow ST_r(w,B))$ from the axioms.

4.11. Example *(Classical Modal Logic).* The key parts of the relational translation for classical modal logic are

$$\begin{aligned} ST_c(w, \Box A) &= \exists V\,(N(w,V) \wedge \forall v\,(v \in V \leftrightarrow ST_c(v,A))) \\ ST_c(w, \Diamond A) &= \forall V\,(N(w,V) \rightarrow \exists v\,(v \in V \leftrightarrow ST_c(v,A))) \end{aligned} \tag{4.6}$$

for the first version (weak semantics), and

$$\begin{aligned} ST'_c(w, \Box A) &= \exists V\,(N(w,V) \wedge \forall v\,(v \in V \rightarrow ST'_c(v,A))) \\ ST'_c(w, \Diamond A) &= \forall V\,(N(w,V) \rightarrow \exists v\,(v \in V \wedge ST'_c(v,A))) \end{aligned} \tag{4.7}$$

for the second version (stronger semantics).

Since the variable V is interpreted as a set and the membership predicate $\in$ has the usual special meaning, this is not yet a suitable first-order translation. It is, however, possible to eliminate the set variables and to replace the membership predicate $\in$ with an uninterpreted binary symbol in. Suppose ϕ is the result of one of the above relational translations of a modal formula. We can add the equivalence

$$\forall w, X\,(N(w,X) \leftrightarrow \exists u\,(N'(w,u) \wedge (\forall v\, in(u,v) \leftrightarrow v \in X))) \tag{4.8}$$

to ϕ. Then, ϕ is satisfiable if and only if $\phi \wedge (4.8)$ is satisfiable. The non-trivial part is to construct a model $\Im'$ for $\phi \wedge (4.8)$ from a model $\Im$ for ϕ. The formulae in ϕ may contain translations of formulae $\Box A$ or $\Diamond A$ (since $\Diamond A \leftrightarrow \neg\Box\neg A$, this case can be reduced to the $\Box$-case).

Applying (4.8) as a rewrite rule to $ST_c(w, \Box A)$ yields:

$$\begin{array}{ll} ST_c(w, \Box A) & \\ \quad = & \exists V\,(N(w,V) \wedge \forall v\, v \in V \leftrightarrow ST_c(v,A)) \\ \quad \leftrightarrow & \exists V\,\exists u\,(N'(w,u) \wedge (\forall v\, in(u,v) \leftrightarrow v \in V) \wedge \forall v\, v \in V \leftrightarrow ST_c(v,A)) \\ \quad \leftrightarrow & \exists V\,\exists u\,(N'(w,u) \wedge (\forall v\, in(u,v) \leftrightarrow v \in V) \wedge \forall v\, in(u,v) \leftrightarrow ST_c(v,A)) \\ \quad \leftrightarrow & \exists u\,(N'(w,u) \wedge (\exists V\,\forall v\, in(u,v) \leftrightarrow v \in V) \wedge \forall v\, in(u,v) \leftrightarrow ST_c(v,A)) \\ \quad \leftrightarrow & \exists u\,(N'(w,u) \wedge \forall v\, in(u,v) \leftrightarrow ST_c(v,A)). \end{array}$$

This way, all occurrences of N can be eliminated from ϕ, and subsequently (4.8) can also be eliminated in a satisfiability-preserving way. N' and in are now uninterpreted binary relations, just as ordinary accessibility relations in normal modal logics. For the logic with the stronger semantics (4.7) we get a similar translation

$$ST'_c(w, \Box A) \quad \leftrightarrow \quad \exists u\,(N'(w,u) \wedge \forall v\,(in(u,v) \rightarrow ST'_c(v,A)))$$

which is, in fact, equivalent to the relational translation of a normal bi-modal logic formula $\langle N' \rangle [in] A$ with two modalities corresponding to the two accessibility relations N' and in. This justifies a translation from classical modal logic (with strong neighborhood semantics) to normal bi-modal logic: $Tr(\Box A) = \langle N' \rangle [in]\, Tr(A)$ (cf. also [Gasquet and Herzig 1993]). Due to the equivalence sign at the right-hand side of (4.6) there is no such simple translation for classical modal logic with weak neighborhood semantics. Nevertheless, the trick with (4.8) yields a translation into FO. Unfortunately, the translation usually yields very large clause sets.

4.12. Remark. The technique for eliminating the set variables and the membership relation is typical for a general technique to modify or improve existing transformations and translations. If A is the result of an existing translation, one adds a definition δ of some relation or function f, occurring in A, to ϕ. δ must define f in terms of some *new* relations or functions, and adding δ must be satisfiability preserving. Once this is proved, one can apply the definition of f to A and eliminate f completely from A. After this, one has to prove that deleting δ is also satisfiability preserving. The net effect is a transformation of A to some A', which, in most cases, can be defined as a single rewrite operation.

In [Baaz, Fermüller and Leitsch 1994] such transformations are called *structural* since they preserve much of the structure of the original input formula; see [Baaz et al. 2001] (Chapter 5 of this Handbook) for the use of abbreviations in the setting of normal form transformations. In Subsection 5.4 we briefly mention their use in decision procedures.

4.13. EXAMPLE *(Quantified Modal Logic).* The standard relational translation for the quantified case can also be derived in a straightforward way. Each flexible function symbol gets an extra argument for the world variable. For a term t, $ST(w,t)$ inserts the variable w into these extra argument positions. For formulating the condition "a in w's domain" in the quantifiers we introduces a relation $in(w,a)$ and translate quantified formulae as

$$\begin{aligned} ST(w,\forall x\, A) &= \forall x\, (in(w,x) \rightarrow ST(w,A)) && (4.9)\\ ST(w,\exists x\, A) &= \exists x\, (in(w,x) \wedge ST(w,A)). && (4.10)\end{aligned}$$

The translation rules for the other connectives and operators are as usual. As an example, take the translation of the well-known *Barcan formula*:

$$\begin{aligned} &ST(w, \forall x \Box p(x) \rightarrow \Box \forall x\, p(x)) = \\ &\quad \forall x\, (in(w,x) \rightarrow \forall v\, (R(w,v) \rightarrow P(v,x))) \rightarrow \\ &\quad\ \forall v\, (R(w,v) \rightarrow \forall x\, (in(v,x) \rightarrow P(v,x)))\end{aligned}$$

The in predicate can be omitted for logics with constant domain interpretation ($in(x,w)$ is always $true$). For increasing domain interpretations, however, the extra axiom $\forall x,w,v\, (in(w,x) \wedge R(w,v) \rightarrow in(v,x))$ is needed, and for decreasing domain interpretations we use the corresponding dual form.

We hope that these examples demonstrate that the standard relational translation can be derived in a straightforward way from the semantics of the connectives and quantifiers in the source logic. Since the translation is almost one-to-one, soundness and completeness proofs are usually straightforward.

Finally, we should point out that it is possible to give a possible worlds semantics for a very general class of logics, thus allowing the formulae of these logics to be translated into classical logic (see [Gabbay 1999, Chapter 2]).

4.3. Expressive power

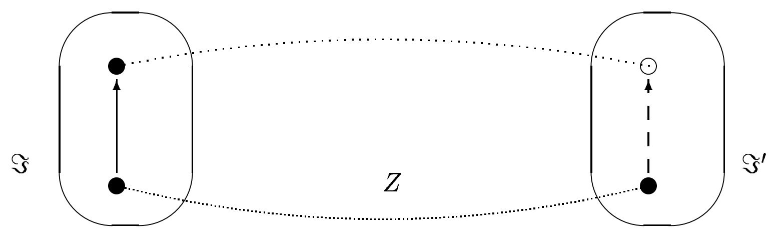
Now that we have introduced the standard relational translation, which identifies modal languages as fragments of first-order or other predicate logics, let us take a closer look at what we have actually obtained. We will first consider matters related to expressive power, and then, in Section 4.4, focus on computing with translations of nonclassical formulae. In particular, we will first look at semantic characterizations of the modal fragment in terms of bisimulations; after that we consider several explanations for the good logical and computational behavior of the modal fragment.

By Theorem 4.2, a modal formula A of the basic modal language can be viewed as a unary FO formula $\alpha(x)$. How much of FO does this modal fragment cover? While this is essentially a *syntactic* question, the best way to answer it is using *semantic* means. The key tool here is provided by the notion of bisimulation.

4.14. Definition. Let $\Im$, $\Im'$ be two interpretations for the basic modal language. A *bisimulation* between $\Im$ and $\Im'$ is a non-empty relation Z between the domain of $\Im$ and the domain of $\Im'$, respectively, that satisfies the following conditions:

1. Z-related states satisfy the same proposition letters;
2. if wZw' and $R(w,v)$ in $\Im$, then there exists $v' \in \Im'$ such that $R'(w',v')$ in $\Im'$ and vZv'; and
3. if wZw' and $R'(w',v')$ in $\Im'$, then there exists $v \in \Im$ such that $R(w,v)$ in $\Im$ and vZv'.

The following figure illustrates items 2 and 3:



Here, the dotted lines indicate the bisimulation relation, the solid and dashed arrows indicates the binary relations R and R'.

If there exists a bisimulation linking two states w and w', then we say that w and w' are *bisimilar*; two models are called *bisimilar* if there exists a bisimulation between them.

4.15. Example. Figure 1 contains two bisimilar interpretations $\Im$ and $\Im'$; the p's indicate that the proposition letter p is true at the state it decorates. We leave it to the reader to show that the dotted lines form bisimulation between $\Im$ and its unfolding $\Im'$.

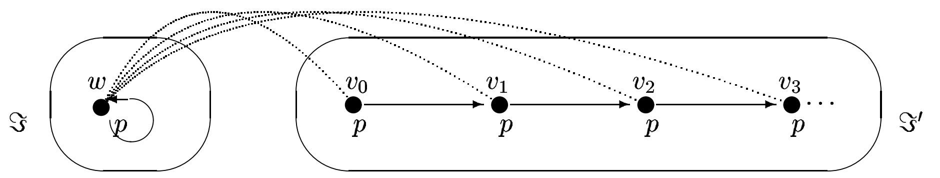


Figure 1: Bisimilar models

Our second example concerns two states w and w' in two interpretations $\mathfrak{J}$ and $\mathfrak{J}'$ such that there cannot be a bisimulation between w and w'. Consider Figure 2; here, each interpretation has a single branch of finite length, and, in addition, $\mathfrak{J}'$ has one branch of infinite length. Because of this infinite branch, there cannot be a bisimulation linking the roots w and w'. Observe that, despite this, w and w' do satisfy exactly the same modal formulae.

The following result explains why bisimulations matter for modal logic.

4.16. PROPOSITION. *Bisimilar models are modally equivalent. More precisely, let $\Im$ and $\Im'$ be two interpretations for the basic modal logic, and let w and w' be two states in $\Im$ and $\Im'$, respectively. Assume that w and w' are bisimilar. Then, for every modal formula A we have that $\Im, w \models A$ iff $\Im, w' \models A$.*

It follows from Proposition 4.16 that all FO formulae in the modal fragment (i.e., unary FO formulae that are in the range of the standard relational translation) are *preserved under bisimulations* in the sense that whenever two interpretations are bisimilar, then we cannot distinguish between them using formulae from the modal fragment. The following result states that this behavior is in fact characteristic of the modal fragment.

4.17. THEOREM. *Let $\alpha(x)$ be a unary first-order formula over a vocabulary with a single binary relation symbol and unary predicate symbols. Then $\alpha(x)$ is (equivalent to) the standard relational translation of a uni-modal formula iff it is preserved under bisimulations.*

PROOF. The 'only-if' direction is just a reformulation of Proposition 4.16. The 'if' direction requires some work. Given a unary FO formula $\alpha(x)$ with the stated property, we have to find a modal formula A such that $ST_m(x, A)$ is equivalent to $\alpha(x)$. The key steps in the proof are as follows.

Consider the set $MC(\alpha) := \{ST_m(x, A) \mid A \text{ is modal}, \alpha(x) \models ST_m(x, A)\}$. It suffices to show that $MC(\alpha) \models \alpha$. To do so, take an interpretation $\Im$ and state and w such that $\Im, w \models MC(\alpha)$; we need to show that $\Im, w \models \alpha$.

First, we use some general tricks from first-order logic to find an interpretation $\mathfrak{J}$ and state v such that $\mathfrak{J}, v \models MC(\alpha) \cup \{\alpha(x)\}$. Then, v (in $\mathfrak{J}$) and w (in $\Im$) satisfy the same modal formulae — but they need not be bisimilar. By some general tricks from first-order logic again, we can massage $\Im, w$ and $\mathfrak{J}, v$ into so-called ω-saturated interpretations $\Im^*, w^*$ and $\mathfrak{J}^*, v^*$, respectively, where it can be shown that the relation of satisfying the same modal formulae is indeed a bisimulation. To wrap up the proof, we chase α from $\mathfrak{J}, v$ to $\mathfrak{J}^*, v^*$ (by first-order logic) to $\Im^*, w^*$ (by preservation under bisimulations) to $\Im, w$ (by first-order logic) — and this is what we needed. $\square$

Theorem 4.17 is originally due to van Benthem [1976]. During the 1990s many differ-

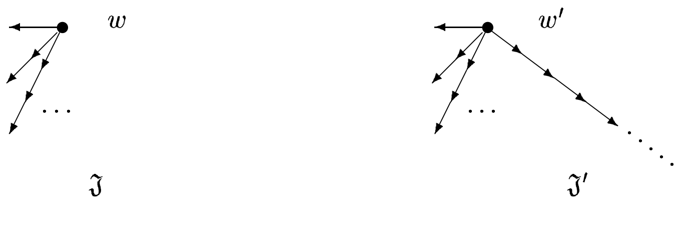


Figure 2: Equivalent but not bisimilar

ent flavors of Theorem 4.17 were established: for the modal mu-calculus [Janin and Walukiewicz 1996], for finite models [Rosen 1997], for temporal logic [Kurtonina and de Rijke 1997], for certain fragments of knowledge representation languages [Kurtonina and de Rijke 1999], for CTL* [Moller and Rabinovich 1999], and for description logics [Areces and de Rijke 2000].

Now that Theorem 4.17 has given us a precise characterization of the modal fragment, let us take a closer look at some of its logical and computational properties. We will see that the fragment behaves very well in many respects, and our main concern will be to try and understand why this is so. To begin with, the modal fragment has the finite model property.

4.18. DEFINITION. A set of formulae X is said to have the *finite model property* if every satisfiable formula in X is satisfiable in a finite interpretation.

The modal fragment, or equivalently, the uni-modal language, has the finite model property in a very strong sense: a modal formula is A satisfiable iff it is satisfiable on an interpretation $\Im$ with at most $2^{|A|}$ states, where $|A|$ is the length of A. *Deciding* whether a modal formula is satisfiable at all is in fact 'easier' than the strong finite model property may lead one to believe: it is a PSPACE-complete problem [Ladner 1977].

What is probably more significant from an automated reasoning point of view is the fact that the modal fragment has the tree model property.

4.19. DEFINITION. A *tree model* for the basic modal language is an interpretation $\Im = (W, R, P)$ where (W, R) is a tree. A set of formulae X is said to have the *tree model property* if every satisfiable formula $A \in X$ is satisfiable at the root of a tree model.

Using the notion of bisimulation (Definition 4.14) it is easy to show that the modal fragment does indeed have the tree model property: let A be a satisfiable modal formula, and let $\Im, w$ be such that $\Im, w \models A$. Let $\mathfrak{J}, w'$ be the complete unfolding of $\Im$ (starting from w). Then, $\mathfrak{J}$ is a tree model, and $\mathfrak{J}$ and $\Im$ are bisimilar via a bisimulation that links w in $\Im$ to w' in $\mathfrak{J}$. Thus, A is satisfied at the root of a tree model, as desired.

What's the importance of the tree model property? First, it paves the way for the use of automata-theoretic tools and tableaux-based decision methods. Moreover, according to Vardi [1997] the tree model property is essential for explaining the *robust decidability* of the modal fragment. This is the phenomenon that the modal fragment is decidable itself, and of reasonably low complexity, and that these features are preserved when the fragment is extended by a variety of additional constructions, including counting, transitive closure, and least fixed points.

So far, we have seen a number of examples of the good logical and computational behavior of the modal fragment: the finite model property, the tree model property, robust decidability, but there are many more — see [de Rijke 1999]. *Why* does the modal fragment enjoy this behavior? Since the early 1980s several answers have been suggested. One of the first was due to Gabbay [1981*a*], who tried to give an explanation in terms of finite variable fragments; these are defined by restricting the number of variables (free or bound) that may be used.

4.20. DEFINITION *(Finite Variable Fragments).* Let k be a natural number. We write FO^k to denote the restriction of first-order logic to formulae of a relational vocabulary (that is, without function symbols) that contain only the variables v_1, …, v_k. Note that interesting sentences in FO^k are not in prenex normal form; on the contrary, one extensively re-uses variables.

4.21. EXAMPLE. To express that a graph $G = (V, E)$ contains a path of length 4, a sentence in prenex normal form needs 5 variables. By re-using variables, the property is expressible in FO^2, by a sentence of the form

$$\exists v_1 \exists v_2\, (E(v_1, v_2) \wedge \exists v_1\, (E(v_2, v_1) \wedge \exists v_2\, (E(v_1, v_2) \wedge \exists v_1\, (E(v_2, v_1))))).$$

Finite variable fragments have a long history. The earliest systematic study seems to be due to Henkin [1967] in the setting of algebraic logic; Gabbay [1981*a*] develops the modal connection, and Immerman and Kozen [1987] study the link with complexity and database theory. We refer the reader to [Otto 1997] for more on FO^k.

What do finite variable fragments have to do with modal logic and with the modal fragment? Gabbay [1981*a*] observed that we only need two variables to carry out the standard relational translation; see Table 4. By way of example, consider the formula $\Diamond\Box\Diamond p$ again; using only two variables, its translation becomes $ST_m(v_1, \Diamond\Box\Diamond p) = \exists v_2\, (R(v_1, v_2) \wedge \forall v_1\, (R(v_2, v_1) \to \exists v_2\, (R(v_1, v_2) \wedge p(v_2))))$.

$$\begin{aligned}
ST_m(v_1, p) &= p(v_1)\\
ST_m(v_1, \neg\phi) &= \neg ST_m(v_1, \phi)\\
ST_m(v_1, \phi \wedge \psi) &= ST_m(v_1, \phi) \wedge ST_m(v_1, \psi)\\
ST_m(v_1, \Diamond\phi) &= \exists v_2\, (R(v_1, v_2) \wedge ST_m(v_2, \phi))\\
ST_m(v_1, \Box\phi) &= \forall v_2\, (R(v_1, v_2) \to ST_m(v_2, \phi))\\
ST_m(v_2, p) &= p(v_2)\\
ST_m(v_2, \neg\phi) &= \neg ST_m(v_2, \phi)\\
ST_m(v_2, \phi \wedge \psi) &= ST_m(v_2, \phi) \wedge ST_m(v_2, \psi)\\
ST_m(v_2, \Diamond\phi) &= \exists v_1\, (R(v_2, v_1) \wedge ST_m(v_1, \phi))\\
ST_m(v_2, \Box\phi) &= \forall v_1\, (R(v_2, v_1) \to ST_m(v_1, \phi))
\end{aligned}$$

Table 4: Two variables suffice

It is easy to see that for all modal formulae A: $\Im, w \models A$ iff $\Im, w \models ST_m(w, A)$. Hence, the modal fragment is really part of FO^2. Does this embedding explain the good behavior of the modal fragment? The first decidability result for FO^2 was obtained by Scott [1962], who showed that the decision problem for FO^2 can be reduced to that of the Gödel class, i.e., the set of first-order formulae with quantifier prefix $\exists^*\forall\forall\exists^*$. Since the Gödel class without equality is decidable [Gödel 1932, Kalmár 1933, Schütte 1934], Scott's reduction yields the decidability of FO^2 without equality. Mortimer [1975] established decidability of FO^2 with equality. In contrast, for $k \geq 3$, FO^k is undecidable. Moreover, FO^k does not have the tree model property for any $k \geq 2$; to see this, simply observe that $\forall x, y\, R(x, y)$ is in FO^2. To make matters worse, FO^2 is not robustly decidable: adding any of counting, transitive closure, or least fixed points results in undecidable systems [Grädel, Otto and Rosen 1997*b*].

In conclusion, then, finite variable fragments don't seem to offer a satisfactory explanation for the good logical and computational properties of the modal fragment. An alternative proposal, launched in the late 1990s stresses the special nature of the quantifiers occurring in formulae in the modal fragment. Guarded logics [Andréka, van Benthem and Németi 1998] are defined by restricting quantification in first-order logic, second-order logic or fixed-point logic in such a way that, semantically speaking, subformulae can only refer to objects that are 'nearby' or 'guarded.' Syntactically, this means that all quantifiers must be relativized by 'guards.'

4.22. DEFINITION *((Loosely) Guarded Fragment).* The *guarded fragment* (GF) is defined as follows

1. Every relational atomic formula $Rx_{i_1} \ldots R_{i_k}$ or $x_i = x_j$ is in GF;
2. GF is closed under boolean connectives
3. GF is closed under guarded quantification: if $\overline{x}$ and $\overline{y}$ are tuples of variables, $\alpha(\overline{x}, \overline{y})$ is an atomic formula, and $A(\overline{x}, \overline{y})$ is a formula in GF such that free$(A) \subseteq$ free$(\alpha) = \{\overline{x}, \overline{y}\}$, then the formulae

$$\exists\overline{x}\, (\alpha(\overline{x}, \overline{y}) \wedge A(\overline{x}, \overline{y})) \text{ and } \forall\overline{x}\, (\alpha(\overline{x}, \overline{y}) \rightarrow A(\overline{x}, \overline{y}))$$

are in GF as well. (Here, free(A) denotes the set of free variables in A.)

The formula $\alpha(\overline{x}, \overline{y})$ in item 3 is called a *guard.*

The *loosely guarded fragment* (LGF) is obtained by relaxing the third condition somewhat, to the following.

3′. LGF is closed under loosely guarded quantification: if $A(\overline{x}, \overline{y})$ is in LGF, and $\alpha(\overline{x}, \overline{y}) = \alpha_1 \wedge \cdots \wedge \alpha_m$ is a conjunction of atomic formulae, then

$$\exists\overline{x}\, (\alpha(\overline{x}, \overline{y}) \wedge A(\overline{x}, \overline{y})) \text{ and } \forall\overline{x}\, (\alpha(\overline{x}, \overline{y}) \rightarrow A(\overline{x}, \overline{y}))$$

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

Clearly, the modal fragment is part of GF. As a further example, the formula $\forall x \forall y\, (R(x, y) \rightarrow R(y, x))$ is in GF, but transitivity $(\forall x \forall y \forall z\, (R(x, y) \wedge R(y, z) \rightarrow$

$R(x,z)))$ is not in GF or in LGF. The formula $\exists y\,(R(x,y) \wedge Py \wedge \forall z\,(R(x,z) \wedge R(z,y) \rightarrow Qz))$ is in LGF, but not in GF.

The guarded fragment behaves much better — both logically and computationally — than finite variable fragments.

4.23. Theorem ([Andréka et al. 1998]). *GF has the (strong) finite model property.*

4.24. Theorem ([Grädel 2000]). *The satisfiability problem for GF is 2EXPTIME-complete. By imposing an upper bound on the arity of the predicates or on the number of variables this may be brought down to EXPTIME-completeness.*

In addition, we have practical decision methods for GF and LGF [de Nivelle 1998, de Nivelle and de Rijke to appear]. As to the *robust* decidability of GF, adding counting, transitivity or functionality statements destroys decidability [Grädel 2000]. Ganzinger, Meyer and Veanes [1999] consider the restriction of the guarded fragment to the two-variable case where, in addition, binary relations may be specified as transitive. This very restricted form of GF without equality is undecidable, but when allowing non-unary relations to occur only in guards, the logic becomes decidable; this latter class contains standard relational translations of modal logics such as **K4**, **S4**, and **S5**. Transitive closure and least fixed points can be added to GF or LGF at no additional computational costs [Grädel and Walukiewicz 1999].

All in all, the guarded fragment seems to offer a much better syntactic explanation for the good logical and computational behavior of the modal fragment than finite variable fragments. But GF is not perfect either: its coverage is too limited to explain the good behavior of, for instance, linear temporal logic (LTL) and systems such as **S4**. Now, LTL can be covered by the loosely guarded fragment LGF [van Benthem 1997], but **S4** is beyond its scope. However, de Nivelle [1999] devised an extended relational translation method, which introduces new relational symbols to translate systems like **S4** into monadic second-order logic; the scope of this method remains to be explored.

Capturing nonclassical logics in terms of guarded or guarded-like fragments continues to be an area of active research.

4.4. Reasoning tasks

We have now got a fairly good understanding of the expressive power of modal fragments. Let us try to exploit the connection between modal and first-order logic, and feed our modal reasoning problems to general first-order automated reasoning tools. Unfortunately, most first-order theorem provers, especially those working with resolution, simply do not constitute decision procedures for (translated) modal formulae. Given an unsatisfiable set of clauses $\mathcal{C}$, any complete resolution procedure will of course generate a contradiction, but the problem is that resolution need not terminate if $\mathcal{C}$ happens to be satisfiable, not even if $\mathcal{C}$ belongs to a decidable fragment (as in the modal case).

4.25. Example. Consider the formula $\Box(p \to \Diamond p)$, which is clearly satisfiable. Proving this amounts to showing that the following set of clauses is satisfiable.

1. $\neg R(c, y) \lor \neg p(y) \lor R(y, f(y))$
2. $\neg R(c, y) \lor \neg p(y) \lor p(f(y))$

The clauses have two resolvents:

3. $\neg R(c, c) \lor \neg p(c) \lor \neg p(f(c)) \lor p(f^2(c))$
4. $\neg R(c, f(y)) \lor R(f(y), f^2(y)) \lor \neg R(c, y) \lor \neg p(y)$.

Clauses 2 and 4 resolve to produce

5. $\neg R(c, f^2(y)) \lor R(f^2(y), f^3(y)) \lor \neg R(c, f(y)) \lor \neg R(c, y) \lor \neg p(y)$.

Clauses 2 and 5 resolve again to produce an analogue of 5 with even higher term-complexity, and so on. None of the clauses is redundant and can be deleted. In the limit our input set has infinitely many resolvents, and this shows that standard 'plain' resolution does not terminate for relational translations of satisfiable modal formulae.

Even for unsatisfiable modal formulae, a 'plain' resolution method is not ideal: the performance may seriously lag behind procedures that have been purpose-built for modal reasoning. Table 5 illustrates the point with a brief comparison between a number of tools. The results in columns 2–4 are taken from [Giunchiglia, Giunchiglia and Tacchella 2000, Horrocks, Patel-Schneider and Sebastiani 2000]; they were obtained on a Pentium 350Mhz with a time out of 100 seconds; the results in column were obtained on a Sun ULTRA II 300MHz, with the same time out. The experiments were performed on 9 collections of tests; each collection consists of 21 provable (in **K**) and 21 unprovable (in **K**) formulae of increasing difficulty; provable formulae are indicated with the 'p' suffix, unprovable ones with the 'n' suffix. If a system could solve all 21 problems, Table 5 lists a $>$ in the relevant cell; the times indicated are times spent on solving the most difficult problems in each category.

The systems mentioned are *SAT 1.2 [Giunchiglia et al. 2000], DLP 3.2 [Patel-Schneider 1998], and TA 1.4 [Hustadt, Schmidt and Weidenbach 1998]. DLP outperforms the other systems on nearly all tests; it is a tableau-based system which implements intelligent backtracking. *SAT also implements intelligent backtracking, and is SAT-based. TA is a translation-based system that does not use the standard relational translation but the more sophisticated functional translation (as discussed in Section 5) on top of various resolution refinements. For comparison, the fifth column, labeled 'Bliksem 1.10a' contains figures obtained by feeding the relational translation to the general purpose FO theorem prover Bliksem [Bli 2000] in auto mode. The experiments, though superficial, seem to indicate that the relational translation combined with plain resolution is simply no match.

What's the cause for the poor performance of general purpose resolution-based tools compared to purpose-built tools? Plain resolution simply does not exploit the special nature of the original modal input formulae — we are 'reducing' a nicely decidable logic to an undecidable one, and, thus far, our reduction does not use, for instance, the finite model property or the tree model property of the modal input.

Roughly speaking, there are two responses to this issue: one which tries to stick to the standard relational translation as we have defined it in Definition 4.1 and

Test		*SAT 1.2		DLP3.2		TA 1.4		Bliksem 1.10a	
		Size	Time	Size	Time	Size	Time	Size	Time
branch	p	>	0.21	19	46.06	6	65.76	3	20.78
	n	12	94.49	13	53.63	6	65.33	3	32.03
d4	p	>	0.06	>	0.05	15	71.11	3	66.57
	n	>	2.87	>	1.12	14	44.06	1	1.65
dum	p	>	0.04	>	0.02	17	64.99	3	55.10
	n	>	0.12	>	0.02	16	65.82	1	4.78
grz	p	>	0.04	>	0.04	>	0.51	5	81.04
	n	>	0.01	>	0.05	>	0.33	0	0.00
lin	p	>	0.01	>	0.03	>	9.24	>	80.57
	n	>	47.80	>	0.13	>	80.01	4	19.86
path	p	>	0.72	>	0.32	5	25.03	4	26.32
	n	>	0.96	>	0.36	4	60.84	2	61.96
ph	p	8	48.54	7	10.23	6	43.16	5	11.33
	n	12	0.60	>	2.69	9	55.13	5	8.21
poly	p	>	1.73	>	0.11	5	53.48	5	70.46
	n	>	2.25	>	0.18	4	9.09	4	52.24
t4p	p	>	0.29	>	0.06	16	88.66	0	0.00
	n	>	1.28	>	0.13	9	87.72	0	0.00

Table 5: Comparison

refines the resolution procedure to combine it with some kind of preprocessing that exploits the special nature of the modal input; the other response is to abandon the translation and use one that somehow encodes more modal information. Sections 5 and 6 below are devoted to a number of alternative translations; the remainder of the present section is devoted to a brief discussion of the former response.

How can resolution be turned into a decision procedure for formulae that we get out of the relational translation? During the 1990s special resolution refinements became available that are aimed at doing just this. Some of this work has been collected in [Fermüller, Leitsch, Tammet and Zamov 1993], which includes a decision procedure for $\mathcal{ALC}$, the description logic counterpart of the modal logic $\mathbf{K}$ that uses orderings plus saturation. We refer the reader to [Fermüller et al. 2001] (Chapter 25 of this Handbook) for more details on resolution-based decision procedures relevant for the (relational) modal fragment.

De Nivelle [1998] has given a resolution decision procedure for GF without equality. In his procedure, a non-liftable ordering is employed, and, as a consequence, additional work is needed for proving refutational completeness; see [de Nivelle and de Rijke to appear] for further details. Ganzinger and de Nivelle [1999] describe a decision procedure for GF with equality which is based on resolution and superposition. More precisely, ordered paramodulation with selection is a decision procedure for GF with equality, and the worst-case time complexity of the decision procedure is doubly-exponential, which is optimal, given that the logic is 2EXPTIME-complete. The procedure can be extended to LGF with equality, but is much more involved there and needs hyper inferences which simultaneously resolve a set of 'guards.'

Instead of devising resolution decision procedures based on the relational translation, an alternative approach is to devise preprocessing techniques that enhance standard resolution by encoding information about the source logic. Areces, Gennari, Heguiabehere and de Rijke [2000] exploit a very strong form of the tree model property to complement the standard relational translation for **K** with additional semantic information. The key idea here is to encode the layering present in tree models into the syntax of modal formulae, by first translating them into an extended multi-modal language where each modal depth has its own modal operators and its own proposition letters, thus avoiding that clauses stemming from different levels will be resolved.

4.26. EXAMPLE. Consider the satisfiable formula $\Box(p \to \Diamond p)$ from Example 4.25 again. By exploiting the tree model property of modal logic it is easy to see that $\Box(p \to \Diamond p)$ is satisfiable iff the following two clauses are:
1. $\neg R_1(c,y) \vee \neg p_1(y) \vee R_2(y, f(y))$
2. $\neg R_1(c,y) \vee \neg p_1(y) \vee p_2(f(y))$.

Here, a literal with subscript i corresponds to a modal operator or proposition letter occurring at modal depth i. Clearly, the set of clauses 1, 2 is saturated — a dramatic improvement over Example 4.25.

To make things precise, we need an intermediate multi-modal language, whose collection of modal operators is $\{\Diamond_i \mid i \geq 0\}$. Let A be a uni-modal formula, and let n be a natural number. The translation $Tr(A,n)$ of A into the intermediate language is defined by

$$\begin{array}{rcl} Tr(p,n) & := & p_n \\ Tr(\neg B,n) & := & \neg\, Tr(B,n) \\ Tr(B \wedge C,n) & := & Tr(B,n) \wedge Tr(C,n) \\ Tr(\Diamond B,n) & := & \Diamond_{n+1}\, Tr(B,n+1). \end{array}$$

The *layered relational translation* $LT(A)$ is the composition of Tr and the relational translation ST_m: $LT(A) = ST_m(w, Tr(A,0))$. The following result may be found in [Areces et al. 2000].

4.27. PROPOSITION. *Let A be a uni-modal formula. Then A is satisfiable in* **K** *iff its layered relational translation $LT(A)$ is.*

To evaluate the net effects of the additional preprocessing step used in the layered relational translation, Areces et al. [2000] carry out tests running the resolution-based prover Bliksem with and without the extra layering on a number of test sets.

For a start, the tests in Table 5 were carried out for the layered translation, leading to significant improvements:

Test		Relational		Layered	
		Size	Time	Size	Time
branch	p	3	20.78	8	76.36
	n	3	32.03	8	70.25
d4	p	3	66.57	11	71.76
	n	1	1.65	6	56.36
dum	p	3	55.10	>	5.01
	n	1	4.78	>	6.45
grz	p	5	81.04	>	12.56
	n	0	0.00	>	53.79
lin	p	>	80.57	>	81.28
	n	4	19.86	5	73.75
path	p	4	26.32	7	50.25
	n	2	61.96	4	26.69
ph	p	5	11.33	5	11.17
	n	5	8.21	5	7.99
poly	p	5	70.46	13	75.76
	n	4	52.24	14	69.36
t4p	p	0	0.00	13	82.77
	n	0	0.00	6	55.94

In addition, test were carried out on problems generated by the modal QBF test set. The latter is the basic yardstick for the TANCS (Tableaux Nonclassical Systems Comparisons) competition on theorem proving and satisfiability testing for nonclassical logics [TAN 2000]. It is a random problem generator that has been designed for evaluating solvers of (un-) satisfiability problems for the modal logic **K**. The formulae of this benchmark are generated using quantified boolean formulae. For unsatisfiable formulae, the use of layering results in a reduction in computing time of up to four orders of magnitude, and an average reduction in the number of clauses generated by one order of magnitude. For satisfiable formulae, similar though less dramatic improvements may be observed.

The layered preprocessing step may be combined with any existing decision procedure. Alternatively, one can devise decision procedures that fully exploit the restricted syntactic nature of the range of the layered relational translation.

To conclude this section, we briefly list what little is known about reasoning with relational encodings of other nonclassical logics. Proving relevance logic theorems with a FO theorem prover using the standard translation is possible. The performance of FO theorem provers without extra strategies, however, is quite poor [Ohlbach and Wrightson 1984]. Moreover, the usual systems do not provide a decision procedure. Unfortunately, not much is known about special strategies for this kind of translated relevance logic formulae, and the same holds true for intuitionistic logic and for classical modal logic. The guarded and loosely guarded fragment provide general first-order fragments into which many nonclassical logics can be embedded via the relational translation and this gives us at least a starting point for computing with such logics. But, clearly, extensive experimental and theoretical work is needed here.

5. The functional translation

In the late 1980s and early 1990s the functional translation approach appeared simultaneously and independently in a number of publications; see [Ohlbach 1988*a*, Fariñas del Cerro and Herzig 1988, Herzig 1989, Zamov 1989, Auffray and Enjalbert 1992]. The functional translation can provide a considerable improvement of the behavior of FO theorem provers on translated nonclassical formulae. In this section we first provide some background material; after that we discuss the expressive power of the functional translation, as well as the notion of prefix stability, which gives rise to an *optimized* functional translation, for which we discuss decidability results in the final subsection.

5.1. Background

The *functional translation* is based on an alternative semantics of modal logic. The fundamental idea is that each binary relation can be defined by a set of (partial or total) functions.

5.1. Proposition. *For any binary relation R on a non-empty set W there is a set AF_R of* accessibility functions*, that is, a set of partial functions $\gamma : W \to W$, such that*

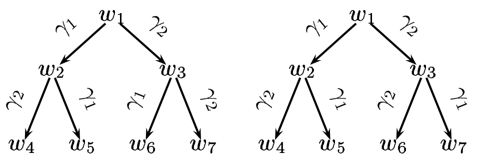
$$\forall x, y\,(R(x,y) \leftrightarrow (\exists\gamma : AF_R\, \gamma(x) = y)). \tag{5.1}$$

Consider the relation R_1 represented by the arrows in the tree below:

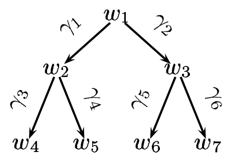
$$\begin{tikzcd}
 & & & w_1 \arrow[dll] \arrow[drr] & & & \\
 & w_2 \arrow[dl] \arrow[dr] & & & & w_3 \arrow[dl] \arrow[dr] & \\
w_4 & & w_5 & & w_6 & & w_7
\end{tikzcd}$$

There are many different ways to decompose R_1 into sets of two functions $\{\gamma_1, \gamma_2\}$. For example:

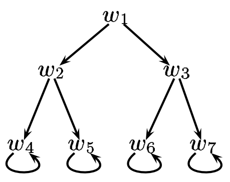
$$\begin{tikzcd}
 & & & w_1 \arrow[dll, "\gamma_1"'] \arrow[drr, "\gamma_2"] & & & \\
 & w_2 \arrow[dl, "\gamma_1"'] \arrow[dr, "\gamma_2"] & & & & w_3 \arrow[dl, "\gamma_1"'] \arrow[dr, "\gamma_2"] & \\
w_4 & & w_5 & & w_6 & & w_7
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & & w_1 \arrow[dll, "\gamma_1"'] \arrow[drr, "\gamma_2"] & & & \\
 & w_2 \arrow[dl, "\gamma_1"'] \arrow[dr, "\gamma_2"] & & & & w_3 \arrow[dl, "\gamma_2"'] \arrow[dr, "\gamma_1"] & \\
w_4 & & w_5 & & w_6 & & w_7
\end{tikzcd}$$

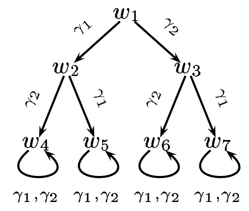


Since the relation is not serial, and there are dead ends, all these γ_1 and γ_2 are partial functions. With partial functions there are even more possible decompositions for a relation. The picture below shows an extreme case where the functions $\gamma_1, \ldots, \gamma_6$ are "as partial as possible".



There is also the other extreme, where each accessibility function is *maximally defined*, i.e., whenever there is some y with $R(x, y)$ then $\gamma(x)$ must be defined. For total (serial) relations it is therefore always possible to decompose the relation into a set of *total* functions. For example, the relation R_2 displayed on the left-hand side can be decomposed into the relation on the right-hand side, where all the functions γ_i are total:





These observations can be exploited for a number of manipulations of formula sets containing a binary relation, and in particular for the relational translation of modal and intuitionistic logics.

Let us make things more formal. First, we need some notation. To avoid quantification over function symbols (in the first-order encoding) as in (5.1), we use a list notation in which any term $\gamma(x)$ is rewritten as $[x\gamma]$. Here, $[\cdot, \cdot]$ denotes the functional application operation which is defined to be a mapping from a domain W to the set of all partial functions over W. So complex terms of the form $\gamma_m(\ldots \gamma_2(\gamma_1(x))\ldots)$ become terms of the form $[[[[x\gamma_1]\gamma_2]\ldots]\gamma_m]$. We usually omit all brackets except the outermost ones, and write $[x\gamma_1\gamma_2\ldots\gamma_m]$ instead of $[[[[x\gamma_1]\gamma_2]\ldots]\gamma_m]$.

Recall that a relation R is *serial* if it satisfies $\forall x \exists y\, R(x, y)$. For serial relations R it is always possible to define R in terms of a set of total functions. For the general case that R is not serial, there can be no set AF_R of *total* functions that 'captures' R. The problem is that the target logic for the functional translation cannot cater for partial functions: $[x\gamma]$ is a first-order term and will always have an interpretation. A standard solution is to adjoin a special element $\bot$ to the domain W of the model at hand, thus obtaining $W^{\bot}$, and to encode every partial function γ as a total function which maps the elements for which it is not defined to the new 'undefined' state $\bot$. Accordingly, any formula in which such an 'undefined' situation occurs, is translated into a conditional formula.

We introduce a special predicate de_R, called the *dead-end* predicate, which is defined by

$$\forall x\,(de_R(x) \leftrightarrow \forall\gamma\,(\gamma \in AF_R \rightarrow [x\gamma] = \bot)) \tag{5.2}$$

5.2. THEOREM. *Let R be a binary relation on a set W, and let $W^{\bot} = W \cup \{\bot\}$. Then, the following defines R in terms of a set AF_R of* total *functions $\gamma : W^{\bot} \rightarrow W^{\bot}$:*

$$\forall x, y : W\,(R(x,y) \leftrightarrow (\neg de_R(x) \wedge \exists\gamma\,(\gamma \in AF_R \wedge [x\gamma] = y))), \tag{5.3}$$

where de_R is defined as in (5.2).

The equivalence (5.3) defines any binary relation R in terms of a set AF_R of total functions and a special set de_R of states. The right-hand side of (5.3) says that if x is not a dead-end in R, then there is a total function γ which maps x to y. If R is serial, then de_R is the empty set, and (5.3) simplifies to (5.1).

We are ready to define the functional semantics for modal logic.

5.3. DEFINITION. A *functional frame* is a 4-tuple $\mathcal{F} = (W, de, AF, [\cdot,\cdot])$, where W is a non-empty set, de is a subset of W, AF is a set of *total* functions $\gamma : W \rightarrow W$, and $[\cdot,\cdot] : W \times AF \rightarrow W$ the functional application operation.

A *functional model* is a pair $\Im = (\mathcal{F}, P)$, where $\mathcal{F}$ is a functional frame, and P is a valuation. The new truth definition for the diamond operator is

$$\Im, w \models \Diamond A \text{ iff } w \notin de \text{ and } \exists\gamma : AF\,(\Im, [w\gamma] \models A),$$

and dually for the box operator.

It can be shown that any modal logic $\mathbf{K\Sigma}$ is complete with respect to a class of relational frames (or models) iff it is complete with respect to a class of functional frames (or models); see, for instance, [Schmidt 1997, Chapter 2].

5.4. DEFINITION *(Functional Translation).* First, we need to specify the target logic. Following [Schmidt 1997] we use a many-sorted logic with a sort hierarchy and set declarations for function symbols [Walther 1987]. In this logic, a sort symbol can be a viewed as a unary predicate and it denotes a subset of the domain.

For the functional translation we introduce the sorts W and AF. The variables x, y, z, ..., are assumed to be of sort W; the functional variables are denoted by γ, γ_1, γ_2, ..., and are or sort AF. The sort of the operator $[\cdot,\cdot]$ is $W \times AF \rightarrow W$.

The *functional translation* $FT(t, A)$ is defined as follows:

$$FT(t,p) \quad = \quad p(t)$$

$$\begin{aligned}
FT(t,\neg A) &= \neg FT(t,A)\\
FT(t,A\vee B) &= FT(t,A)\vee FT(t,B)\\
FT(t,A\wedge B) &= FT(t,A)\wedge FT(t,B)\\
FT(t,\Diamond A) &= \begin{cases}\exists\gamma : AF\ FT([t\gamma],A), & \text{if } \Diamond \text{ is serial}\\ \neg de(t)\wedge\exists\gamma : AF\ FT([t\gamma],A), & \text{otherwise.}\end{cases}\\
FT(t,\Box A) &= \begin{cases}\forall\gamma : AF\ FT([t\gamma],A), & \text{if } \Box \text{ is serial}\\ \neg de(t)\rightarrow\forall\gamma : AF\ FT([t\gamma],A), & \text{otherwise.}\end{cases}
\end{aligned}$$

Here, γ is assumed to be a variable not occurring in t.

The functional translation can easily be extended to multi-modal logic. For each modality M one introduces an extra sort AF_M.

5.5. Example. Take the McKinsey formula $\Box\Diamond p \rightarrow \Diamond\Box p$. Its functional translation is

$$\forall x\,(\neg de(x) \rightarrow \forall\gamma(\neg de([x\gamma])\wedge\exists\delta\, p([x\gamma\delta])) \rightarrow (\neg de(x)\wedge\exists\gamma(\neg de[x\gamma] \rightarrow \forall\delta\, p([x\gamma\delta])))).$$

5.6. Example *(Wise Men Puzzle).* This example is a famous puzzle used to illustrate the application of modal logic in formalizing the notion of *knowledge*.

> A certain king wishes to determine which one of his three wise men is the wisest. He arranges them in a circle so that they can see and hear each other and tells them that he will put a white or black spot on each of their foreheads, but at least one spot will be white. (In fact all three spots are white.) He offers his favor to the one who first tells him the color of his own spot. After a while, the wisest announces that his spot is white. How does he know?

In our formalization we introduce constant symbols a, b, c to name the three wise men. They are rigid symbols, not depending on the worlds. Modal operators $[a]$, $[b]$ and $[c]$ are used to encode the notion "wise man a (b, c) *knows*." For a variable y, the operator $[y]$ means "wise man y knows." In addition, we use the standard $\Box$-operator to encode "everybody knows." The modality associated with this $\Box$ is of **S5**-type; it quantifies over all worlds. The predicate $WS(a)$ means "a has a white spot." WS is the only flexible symbol in this example. Let c be the wisest of the three men. The axioms relevant for solving this puzzle are:

- $a \neq b \wedge a \neq b \wedge b \neq c$ (All three wise men are different.)
- $\Box(WS(a) \vee WS(b) \vee WS(c))$ (Everybody knows that one of them has a white spot.)
- $\Box(\forall x, y\, x \neq y \rightarrow (\neg WS(x) \rightarrow [y]\neg WS(x)))$ (They can see each other. Therefore they know when any other one has no white spot.)
- $[c][b]\neg[a]\ WS(a)$ (c knows that b knows that a is not aware of his white spot.)
- $[c]\neg[b]\ WS(b)$ (c knows that b is not aware of his white spot.)

The theorem to prove is $[c]\ WS(c)$ (c knows the color of his own spot).

We use this example also to illustrate some optimizations of the functional translation. An optimized functional translation and clausification of the axioms and the *negated* theorem is:

(1) $b \neq a$
(2) $c \neq a$
(3) $c \neq b$
(4) $WS(w, a) \vee WS(w, b) \vee WS(w, c)$[2]
(5) $x = y \vee WS(w, x) \vee \neg WS([w\gamma_y], x)$[3]
(6) $\neg WS([w_0\gamma_{1c}\gamma_{2b}d_a], a)$[4]
(7) $\neg WS([w_0\gamma_{1c}d_b], b)$
(8) $\neg WS([wd_c], c)$.[5]

A sequence of UR resolutions proves the theorem:

(9)	$[7, 5, 1]$	$\neg WS([w_0\gamma_{1c}d_b\gamma_a], b)$[6]	$([c]\langle b\rangle[a]\neg WS(b))$[7]
(10)	$[9, 4, 6]$	$WS([w_0\gamma_{1c}d_b dd_a], c)$[8]	$([c]\langle b\rangle\langle a\rangle\ WS(c))$[9]
(11)	$[10, 5, 2]$	$WS([w_0\gamma_{1c}, d_b], c)$	$([c]\langle b\rangle\, WS(c))$[10]
(12)	$[11, 5, 3]$	$WS([w_0\gamma_{1c}], c)$	$([c]\, WS(c))$
(13)	$[12, 8]$	empty clause.	

The derivations obviously represent nontrivial conclusions. A more intuitive and comprehensible explanation requires a lot more intermediate steps.

The following result states that the functional translation is sound and complete.

5.7. Theorem. *Let* $\mathbf{K\Sigma}$ *be a modal logic that is complete with respect to a class of relational models. A modal formula* A *is a theorem of* $\mathbf{K\Sigma}$ *iff the conjunction of the following formulae is unsatisfiable*

1. $\forall \overline{p} \forall x\, FT(x, \Sigma)$
2. $\neg\forall x\, FT(x, A)$
3. (5.3).

[2]The variable w originates from the $\Box$-operator which in this case quantifies over all worlds. Therefore it is still a "world variable," not a accessibility function variable.

[3]The variable w again originates from the $\Box$-operator. The variable y ranges over the three wise men. The term γ_y denotes an accessibility function for the knowledge operator for wise man y.

[4]w_0 is a Skolem constant originating from the negated universal quantifier for the negated theorem. It denotes the world where the contradiction manifests itself if the unnegated theorem is in fact universally valid. The terms in the world path $\gamma_{1c}\gamma_{2b}d_a$ have to be written this way in order to match γ_y in the axiom above; γ_{1c} stems from $[c]$, γ_{1b} stems from $[b]$ and d_a is a Skolem constant that stems from $\neg[a]$.

[5](8) is the translated negated theorem. d_c is the Skolem constant stemming from the negated $[c]$-operator.

[6]The variable w in (5) ranges over all worlds. It can therefore be instantiated with whole world paths (this requires a theory unification procedure). The unifier for this step is actually $\{x \mapsto b, y \mapsto a, w \mapsto w_0\gamma_{1c}d_b\}$.

[7]c knows that b considers that a knows that b does not know the color of his spot.

[8]The unifier for this step is $\{\gamma \mapsto b, w \mapsto w_0\gamma_{1c})d_b d_a\}$.

[9]c knows that b considers that a considers that c's spot is white.

[10]c knows that b considers that c's spot is white.

5.2. Frame properties

Many variants of modal logics and other nonclassical logics are characterized by properties of the accessibility relation. If these properties are first-order axiomatizable, the corresponding axioms can easily be added to the standard translation. For the functional translation, the axioms must be translated into the functional language. This is usually straightforward, but the translated axioms are in general equations. In some cases one can simplify the resulting equations again by pulling the existential quantifier over the universal quantifiers.

Let us have a look at some examples; consult Table 6 for a summary.

5.8. EXAMPLE. We use the "$[x\gamma]$" notation. The functional counterpart of *reflexivity* (the modal logic **T**) is:

$$\forall x \exists \gamma : AF\, [x\gamma] = x$$

In maximal functional frames one may pull the existential quantifier to the front:

$$\exists \gamma : AF \forall x\, [x\gamma] = x.$$

Skolemization then yields $\forall x\, [x\, id] = x$, where id is the identity function on worlds. The result is quite intuitive. It means that every world can be mapped to itself by an accessibility function, and this is reflexivity.

5.9. EXAMPLE. On serial frames, the functional version of the *symmetry* axiom $p \to \Box\Diamond p$ (modal logic **B**) is

$$\forall x \forall \gamma_1 \exists \gamma_2\, (x = [x\gamma_1\gamma_2])$$

(If γ_1 takes some world x to some world y then γ_2 takes y back to γ_1; γ_2 is some kind of inverse function to γ_1. In tree-like frames, it really is the inverse function.)

5.10. EXAMPLE. For the transitivity axiom $\Box p \to \Box\Box p$ (modal logic **K4**) we obtain

$$\forall x \forall \gamma_1, \gamma_2 \exists \gamma\, ((\neg de(x) \wedge \neg de([x\gamma_1])) \to [x\gamma_1\gamma_2] = [x\gamma])$$

(In words: γ is the composition of γ_1 and γ_2. The equation states that the composition of accessibility functions is again an accessibility function, mapping worlds to accessible worlds.)

5.3. Prefix stability and the optimized functional translation

The clause form of functionally translated formulae may contain Skolem functions. These may cause complex terms to be built up during resolution. It can be shown that at least for the case of propositional modal logics, Skolem constants are sufficient. This means that functionally translated formulae in clausal form are function

name	axiom	functional property (without seriality)
D	$\Box A \to \Diamond A$	$\forall x\, \neg de(x)$
T	$\Box A \to A$	$\forall x \exists \gamma\, (\neg de(x) \wedge x = [x\gamma])$
B	$A \to \Box\Diamond A$	$\forall x \forall \gamma \exists \delta\, ((\neg de(x) \to \neg de([x\gamma])) \wedge (\neg de(x) \to x = [x\gamma\delta]))$
4	$\Box A \to \Box\Box A$	$\forall x \forall \gamma_1, \gamma_2 \exists \gamma\, ((\neg de(x) \wedge \neg de([x\gamma_1])) \to [x\gamma_1\gamma_2] = [x\gamma])$
5	$\Diamond A \to \Box\Diamond A$	$\forall x \forall \gamma \forall \delta \exists \epsilon\, ((\neg de(x) \to \neg de([x\delta])) \wedge (\neg de(x) \to [x\gamma] = [x\delta\epsilon]))$

Table 6: Axioms and their functional frame properties

free (if world paths are taken as sequences of variables and constants), and this simplifies the development of terminating resolution strategies. The present and the next subsection are devoted to an explanation of the underlying machinery. The first concept that we need to understand is *prefix stability.*

Skolem functions can be avoided if it is possible to pull existential quantifiers originating from $\Diamond$-operators over universal quantifiers originating from $\Box$-operators. The following formula illustrates what this means.

$$\begin{aligned} FT(\Box\Diamond p) \quad := \quad \forall w\, FT(w, \Box\Diamond p) \quad &= \quad \forall w \forall \gamma_1 \exists \gamma_2\, p([w\gamma_1\gamma_2]) \\ &\Leftrightarrow \quad \forall w \exists \gamma_2 \forall \gamma_1\, p([w\gamma_1\gamma_2]). \end{aligned}$$

From a predicate logic point of view, this equivalence does not hold. In general, we are not allowed to pull existential quantifiers arbitrarily over universal quantifiers. Syntactic properties of functionally translated formulae and certain semantic features of "functional frames," however, do indeed guarantee both directions of the equivalence. The syntactic property is "prefix stability" [Ohlbach 1988*a*] (or "unique prefix property" [Auffray and Enjalbert 1992]).

5.11. Definition. Consider a term $t = [x\gamma_1\gamma_2 \ldots \gamma_i\gamma_{i+1} \ldots \gamma_m]$ in the target logic of the functional translation. Any subterm x or $[x\gamma_1 \ldots \gamma_i]$ (for $1 \le i \le m$) is a *prefix in the term t*. The *prefix of a variable* γ_{i+1} in the term t is the term $[x\gamma_1 \ldots \gamma_i]$.

Let T be a set of terms of the form t as above. T is said to be *prefix stable* if any variable γ of type AF occurring in T has exactly one prefix. That is: for any γ, the set $\{s \mid [s\gamma \ldots] \text{ occurs in } T\}$ is a singleton.

5.12. Proposition. *Let A be a modal formula. The set of terms occurring in $FT(w, A)$ is prefix stable.*

Proof. Consider a term $[w\gamma_1 \ldots \gamma_n\gamma]$, occurring somewhere in a functionally translated modal formula; the prefix of γ is $[w\gamma_1 \ldots \gamma_n]$. Of course, γ originates from a particular modal operator M, and $\gamma_1 \ldots \gamma_n$ originate from the nested sequence of modal operators preceding M on the path in the formula tree leading to M. Hence all other occurrences of γ have exactly the same prefix $w\gamma_1 \ldots \gamma_n$. $\Box$

Because prefix stability of terms is such a fundamental concept, Schmidt [1997] gives an independent definition of a logic, called basic non-optimized path logic, that emphasizes the particular ordering of the variables occurring in a literal.

5.13. DEFINITION. The language of *basic non-optimized path logic* is that of monadic first-order logic extended with a non-associative binary operation $[\cdot,\cdot]$ and a designated constant $[]$ which plays the role of the initial world variable w in our functional translation. The sorts are W and AF. There are finitely many unary predicate symbols p, q, $\ldots$, and possibly also a special unary predicate de. The constant $[]$ has sort W, and the function $[\cdot,\cdot]$ maps pairs of world terms and functional terms to world terms. Terms are of the form

$$[[[[[]\gamma_1]\gamma_2]\ldots]\gamma_m] \text{ or in shorthand notation } [\gamma_1\gamma_2\ldots\gamma_m].$$

An atomic basic non-optimized path formula over an ordered set of variables $X_i = \{\gamma_1,\ldots,\gamma_i\}$ is a monadic literal with an argument, as in $p([\gamma_1\ldots\gamma_i])$ or $\neg p([\gamma_1\ldots\gamma_i])$. Complex formulae are defined by induction:

1. Any atomic basic non-optimized path formula over X_i is a basic non-optimized path formula over X_i.
2. $\exists\gamma_{i+1}\,A$ and $\forall\gamma_{i+1}\,A$ are basic non-optimized path formulae over X_i, whenever A is a basic non-optimized path formula over X_{i+1}.
3. Any boolean combination of basic non-optimized path formulae over X_i is a basic non-optimized path formula over X_i.

A sample basic non-optimized path formula over X_3 is the formula

$$\exists\gamma_1\,(\forall\gamma_2\exists\gamma_3\,p([\gamma_1\gamma_2\gamma_3]) \wedge \forall\gamma_2\gamma_3\,\neg p([\gamma_1\gamma_2\gamma_3])).$$

5.14. PROPOSITION. *Let A be a basic non-optimized path formula. The set of terms occurring in A is prefix stable.*

5.15. PROPOSITION. *Let A be a modal formula. Then $FT(w,A)$ is equivalent to a basic non-optimized path formula.*

5.16. PROPOSITION. *Let A be a basic non-optimized path formula. Then there exists a modal formula B such that A is equivalent to $FT(w,B)$.*

The results above establish a direct correspondence between **KD**-formulae and basic non-optimized path formulae. What about non-serial logics? The modal logic **K** can be translated into the modal logic **KD** adjoined with a special propositional symbol de. The translation $(\cdot)^*$ from **K** to **KD** is defined by

$$\begin{aligned} p^* &= p \\ (\neg A)^* &= \neg A^* \\ (A \wedge B)^* &= A^* \wedge B^* \\ (\Diamond A)^* &= \neg de \wedge \Diamond A^* \end{aligned}$$

Informally, the new propositional symbol *de* has the same interpretation as the dead-end predicate of the functional translation: it denotes the set of states at which the accessibility relation is not defined.

5.17. Proposition. *Let A be a modal formula. Then A is provable in* **K** *iff A^* is provable in* **KD**.

The *optimized* functional translation of a modal formula A is obtained by a sequence of two transformations: (1) the functional translation FT as discussed in the preceding subsections, and (2) the quantifier exchange operator Υ, which swaps quantifiers according to the following principle:

$$\exists\gamma\forall\delta\, A \leftrightarrow \forall\delta\exists\gamma\, A. \tag{5.4}$$

The crucial argument which allows one to prove that existential quantifiers over accessibility functions can be pulled over universal quantifiers is of a semantic nature. There are in general many different ways to decompose a binary relation into sets of accessibility functions. We are interested in those functional frames which justify moving existential quantifiers over universal quantifiers. Let us illustrate the basic idea with an example.

Suppose a formula $\Box\Diamond p$ is true at the state w_1, i.e., $w_1 \models \Box\Diamond p$. This means that for every world w accessible from w_1 there is a world accessible from w where p is true. In terms of the relation symbol R this reads as $\forall w\,(R(w_1, w) \Rightarrow \exists v\,(R(w, v)\ \&\ v \models p))$. Consider the situation in the diagram on the left-hand side below. In this model the formula $\exists v\forall w\,(R(w_1, w) \Rightarrow R(w, v)\ \&\ v \models p)$ in which we have swapped the existential quantifier $\exists v$ with the universal quantifier $\forall w$ is false.

$$\begin{tikzcd}[column sep=small]
 & & & w_1\ \Box\Diamond p \arrow[dll] \arrow[drr] & & & \\
 & w_2\ \Diamond p \arrow[dl] \arrow[dr] & & & & w_3\ \Diamond p \arrow[dl] \arrow[dr] & \\
w_4 & & w_5 & & w_6 & & w_7 \\
\neg p & & p & & p & & \neg p
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small]
 & & & w_1 \arrow[dll, "{1,3,5,7}"'] \arrow[drr, "{2,4,6,8}"] & & & \\
 & w_2 \arrow[dl, "{8,6,3,1}"'] \arrow[dr, "{2,4,5,7}"] & & & & w_3 \arrow[dl, "{8,5,4,1}"'] \arrow[dr, "{2,3,6,7}"] & \\
w_4 & & w_5 & & w_6 & & w_7 \\
\neg p & & p & & p & & \neg p
\end{tikzcd}$$

But now consider the functional frame in the diagram on the right-hand side. The numeric labels i denote the accessibility functions γ_i. In the functional language we can express the fact that $\Box\Diamond p$ is true at w_1 by $\forall\gamma\exists\delta\ \delta(\gamma(w_1)) \models p$. In this model we can swap the $\exists\delta$ quantifier and the $\forall\gamma$ quantifiers. $\exists\delta\forall\gamma\,\delta(\gamma(w_1)) \models p$ is true, because the function γ_4 (as well as the function γ_5) maps the worlds w_2 and w_3 to a world where p holds. Moreover, regardless of which one of the worlds w_4, w_5, w_6 or w_7 p is true, in this model there is always a function γ_i which maps w_2 and w_3 to the right worlds.

It can be shown that in functional frames which are *maximal*, i.e., which contain *all* maximally defined accessibility functions, moving the existential quantifiers in front of the universal quantifiers is always justified [Ohlbach and Schmidt 1997]. More precisely, define the *functional extension* of a serial interpretation $\Im$ =

(W, R, P) to be the structure $\Im^* = (W, R, AF, [\cdot, \cdot], P)$, where AF is the *largest* set of all functions that define R.

5.18. THEOREM. *Let A be a modal formula true in a relational interpretation $\Im$. Let $\Im^*$ be its functional extension. For the functional translation B of A the following equivalence is true in $\Im^*$:*

$$\forall\gamma\exists\delta\, B([xt\gamma t'\delta t'']) \leftrightarrow \exists\delta\forall\gamma\, B([xt\gamma t'\delta t'']).$$

The so-called *quantifier exchange operator* Υ converts a non-optimal path formula into prenex normal form and moves *all* quantifiers of functional variables inward as far as possible according to the rule

$$\exists\gamma\forall\delta A \text{ becomes } \forall\delta\exists\gamma A.$$

For any modal formula A, $\Upsilon FT(w, A)$ has a quantifier prefix consisting of a universally quantified world variable followed by a sequence of universally quantified variables of sort AF, and a sequence of existentially quantified variables of sort AF. The quantifier prefix of the negation $\neg\Upsilon FT(w, A)$ is then a sequence of existential quantifiers followed by a sequence of universal quantifiers.

5.19. EXAMPLE. Consider the McKinsey formula $\Box\Diamond p \to \Diamond\Box p$. Recall that the D axiom $\Box p \to \Diamond p$ is a theorem of $\mathbf{K}$ plus the McKinsey formula. The functional translation is given by

$$\forall x\, (\forall\gamma\exists\delta\, p([x\gamma\delta]) \to \exists\gamma'\exists\delta'\, p([x\gamma'\delta'])).$$

The prefix normal form is

$$\forall x\exists\gamma\forall\delta\exists\gamma'\forall\delta'\, (p([x\gamma\delta]) \to p([x\gamma'\delta'])).$$

Applying Υ produces

$$\forall x\forall\delta\forall\delta'\exists\gamma\exists\gamma'\, (p([x\gamma\delta]) \to p([x\gamma'\delta'])).$$

And the negation $\neg\Upsilon\forall FT(x, \Box\Diamond p \to \Diamond\Box p)$ is given by

$$\exists x\exists\delta\exists\delta'\forall\gamma\forall\gamma'\, (p([x\gamma\delta]) \wedge p([x\gamma'\delta'])).$$

Applying Υ to a formula A results in a weaker formula A', since $A \to A'$, for in general $\exists x\forall y\, B$ implies $\forall y\exists y\, B$, but not conversely. The next result provides conditions under which working with the weaker form does suffice to prove A.

5.20. PROPOSITION. *Let $\mathbf{K(D)\Sigma}$ be a complete (or complete and first-order definable) modal logic. Then, for any modal formula A, A is a theorem of $\mathbf{K(D)\Sigma}$ iff $\forall x\, FT(x, \Sigma) \to \Upsilon\forall x\, FT(x, A)$ is a second-order (or first-order) theorem.*

5.21. Proposition. *Let* $\mathbf{K}(\mathbf{D})\mathbf{\Sigma}$ *be a complete (or complete and first-order definable) modal logic with modus ponens and necessitation as its only proof rules. Then, for any modal formula A, A is a theorem of* $\mathbf{K}(\mathbf{D})\mathbf{\Sigma}$ *iff* $\Upsilon\forall x\, FT(x,\Sigma) \rightarrow \Upsilon\forall x\, FT(x,A)$ *is a second-order (or first-order) theorem.*

The functional language is more expressive than the relational language, and properties which are second-order in the relational language may become first-order in the functional language. We should stress that the above results only state that a formula A is a theorem of $\mathbf{K\Sigma}$ iff a weaker theorem follows from weaker frame properties. The net effect of moving existential quantifiers over universal quantifiers in the functional translation of modal formulae is that complex Skolem terms are avoided; at most Skolem constants occur in the resulting clausal forms. In the following subsection we will reap the rewards of this fact. First, however, we will introduce *basic path logics.*

5.22. Definition. *Path logics* are clausal logics. Clauses of basic path logic have the form $p([c\delta e]) \vee \neg q([c\kappa\lambda])$, and are built from constant symbols, like c, e, variables like δ, κ, λ, a special constant symbol $[]$ (denoting the empty path), a left associative operation $[\cdot,\cdot]$, and unary predicate symbols, like p, q, as well as $\vee$ and $\neg$.

The only Skolem terms in basic path clauses are constants. Terms, like $[c\delta e]$ and $[c\kappa\lambda]$, are called paths, and they are required to satisfy prefix stability for variables.

A *path formula* is a conjunction of basic path clauses. *Non-basic path logic* arises when we allow non-empty theories $\Upsilon\forall x\, FT(x,\Sigma)$.

Examples of non-basic path logics are provided by the theories associated with the modal logics $\mathbf{T}$ and $\mathbf{K4}$; their Skolemized formulations are

(right identity) $[x\, id] = x$, where *id* is the identity function, and

(associativity) $[x(\gamma \circ \delta)] = [x\gamma\delta]$, where $\circ$ is functional composition.

5.23. Proposition. *Let* C *be an operator on first-order formulae such that* C(A) *is a clausal form of A. Let A be a modal formula. The set of clauses* $\mathrm{C}(\neg\Upsilon\forall x\, FT(x,A))$ *is a well-formed expression in (basic) path logic, provided that the operation* Υ *moves all existential functional quantifiers inward over all universal quantifiers.*

5.4. Decidability of path logic

In this subsection we discuss decidability aspects of path logics, and hence, by Proposition 5.23, of modal logics. The core idea underlying a resolution-based decision proof for path logic goes back to Joyner Jr. [1976]: let S be a set of path clauses generated by saturation from a finite input set, and establish the existence of a *term depth bound* for terms in S and a bound on the *size* of any clause in S.

Inference for basic path logic may be performed using resolution with syntactic unification. Now, standard equational reasoning with equations such as (right identity) and (associativity) is not very efficient. Therefore, it is usually better to incorporate the equations into suitable theory unification algorithms for the world paths, and, hence, to do inference for non-basic path logics with theory resolution.

In [Ohlbach 1988*b*] unification algorithms for combinations of the above two properties have been presented and proved to be terminating, sound and complete. The proof relies heavily on the prefix stability of world-paths. The algorithm is presented in a Martelli-Montanari style as a number of transformation rules for sets of equations:

(Decomposition)	$f(s_1, \dots, s_n) = f(t_1, \dots, t_n) \to s_1 = t_1 \mathbin{\&} \dots \mathbin{\&} s_n = t_n$
(Separation)	$s\,\mathbf{s} = t\,\mathbf{t} \to s = t \mathbin{\&} \mathbf{s} = \mathbf{t}$
(Identity)	$\mathbf{s}\,w\,\mathbf{s}' = t \to w = [] \mathbin{\&} \mathbf{s}\,\mathbf{s}' = t$
(Inverse)	$\mathbf{s}\,s\,w\,\mathbf{s}' = t \to w = s^{-1} \mathbin{\&} \mathbf{s}\,\mathbf{s}' = t$
(Path-Separation)	$w\,\mathbf{s} = \mathbf{t}\mathbf{t}' \to w = \mathbf{t} \mathbin{\&} \mathbf{s} = \mathbf{t}'$
(Splitting)	$w\,s\,\mathbf{s} = \mathbf{t}\,t\,v\,\mathbf{t}' \to v = v_1\,v_2 \mathbin{\&} w = \mathbf{t}\,t\,v_1 \mathbin{\&} s\,\mathbf{s} = v_2\,\mathbf{t}'.$

Boldface letters represent longer world-paths, whereas normal letters represent single terms. The "Decomposition" and "Separation" rules are sufficient for modal logics **K** and **KD**. The "Identity" rule encodes the reflexivity axiom. It instantiates a variable with an empty world path. The "Inverse" rule encodes the symmetry axiom. It goes together with a simplification rule: $xx^{-1} = []$ and exploits the fact that the prefix of variables w is always the same. Therefore, all occurrences of sw which are instantiated with $w \mapsto s^{-1}$ can be simplified to [] in the same way. The last two rules, "Path-Separation" and "Splitting" encode the transitivity axiom. They are very similar to the corresponding unification rules for associative functions [Auer 1992]. But due to the prefix stability, the application of these rules terminates, in contrast to the rules for associative function symbols.

In the basic case of the modal logics **K** and **KD** there is at most one most general unifier for world-paths. In all other cases there are finitely but sometimes exponentially many most general unifiers. The application of the rules becomes nondeterministic, requiring a tree-like representation of the intermediate steps in the unification. The unification algorithm is no longer minimal, and the same solutions may be computed repeatedly. Schmidt [1998*a*] presents improved unification rules for transitivity, where the world paths are unified mainly from left to right, thus reducing the branching rate of the unification tree.

The optimized functional translation of propositional modal logic formulae yields clauses without function symbols in the world paths. Moreover, instantiation with unifiers computed using the above unification rules except "Path-Separation" and "Splitting" does not increase the length of world-paths. Hence, resolution and factorization do not increase the size of the literals. As a consequence, the number of condensed[11] and not subsumed clauses remains finite.

5.24. Theorem ([Schmidt 1998]). *Any complete resolution procedure with condensing is a decision procedure for the satisfiability problem of a finite set of finite clauses of the basic path logic.*

[11]A clause is *condensed* iff it is not equivalent to one of its factors.

The proof of the above result uses the fact that, for basic path logic, there is no growth of terms, that the basic path logic is closed under resolution with condensing, and that there is no unbounded growth of the size of clauses.

The result can be formulated more generally, as follows.

5.25. Theorem. *Any complete resolution procedure with condensing and possibly a normalization function is a decision procedure for the satisfiability problem of finite clauses in path logics, provided*

1. *a term depth bound exists,*
2. *unification is finitary, and*
3. *the normalization function is effective and returns basic path clauses.*

Although any input set is a set of basic path clauses without any non-constant occurrences of functional terms, the third condition in the above theorem is important as theory unification (needed to handle the theories built into non-basic path logic) may introduce non-constant functional terms.

The first condition in Theorem 5.25 can be interpreted in two ways. First, a term depth bound exists for the particular resolution procedure. Or, an a priori term depth bound exists for the particular path logic; the latter can, for instance, be extracted from the finite model property of the logic. The given term depth can then be used in a simple blocking mechanism to stop the proof procedure from generating clauses whose depth is larger than the given value.

Theorem 5.25 describes the class of path logics (in general, with a theory) for which unrestricted resolution plus condensing is guaranteed to terminate. In particular, the theorem provides a decidability result for the satisfiability problem for those modal logics that can be embedded in path logics such that the three conditions in Theorem 5.25 are met.

5.26. Theorem. *Resolution and condensing (combined with any compatible refinement strategies) is a decision procedure for*

1. *basic path logic and for (the optimized functional translation of)* **K**, **KD**, **S5** *and the multi-modal versions of* **K** *and* **KD***;*
2. *(the optimized functional translation of)* **KT**, *and*
3. **KD4** *and* **S4**.

For practical purposes the solution of using an artificial term depth (as provided by e.g., the finite model property) is poor [Hustadt et al. 1998]. Hence, the search space is very large, even for small input formulae. Therefore, from an efficiency point of view, good search strategies are still badly needed. But due to the above results, any fair strategy is automatically complete.

5.5. Extensions and variations

The idea behind the functional translation is not restricted to logics with ordinary binary accessibility relations, and in this subsection we consider the functional translation of a number of logics other than uni-modal logic.

To begin with, what's the functional translation of quantified modal logic? Modal and intuitionistic formulae that have been translated with the relational translation have a particular structure which can be exploited to get rid of the introduced binary relations. For operators of "universal force" the standard translation yields formulae of the kind $\forall v\,(R(w,v) \to A(v))$ (cf. (4.1), (4.4), (4.5), (4.9)), whereas operators of existential force are turned into $\exists v\,(R(w,v) \wedge A(v))$ (cf. (4.2), (4.10)). If it is known that the relation R is serial, which is the case for the *in* relation introduced by the standard translation (4.9), (4.10) for quantifiers,[12] then Definition 5.4 can be applied to the translated operators of universal force, and translated operators of existential force can be modified in a similar way:

$$\begin{aligned} FT(w, f(t_1,\ldots,t_n)) &= f(w, FT(w,t_1),\ldots,FT(w,t_n)) && (5.5)\\ &\ \text{if } f \text{ is a function or predicate symbol} \\ FT(w, \forall x\ A(x)) &= \forall\delta : AF_{in}\ FT(w,A)[x \mapsto [w\delta]]^{13} && (5.6)\\ FT(w, \exists x\ A(x)) &= \exists\delta : AF_{in}\ FT(w,A)[x \mapsto [w\delta]]. && (5.7)\end{aligned}$$

5.27. EXAMPLE. An example for the functional translation of a quantified modal formula (assuming a serial accessibility relation, but arbitrary domains) is

$$FT(w, \Diamond\Diamond\,(\forall x\,(\Diamond p(x) \wedge \Box q(x)) \to \Diamond(\forall y\, p(y) \wedge \forall z\, q(z)))) =$$

$$\exists\gamma_1,\gamma_2 : AF_R \left[\begin{array}{l} \forall\delta_1 : AF_{in} \left[\begin{array}{l} \exists\gamma_3 : AF_R\, p([w\gamma_1\gamma_2\gamma_3],[w\gamma_1\gamma_2\delta_1]) \wedge \\ \forall\gamma_4 : AF_R\, q([w\gamma_1\gamma_2\gamma_4],[w\gamma_1\gamma_2\delta_1]) \end{array}\right] \\ \to \exists\gamma_5 : AF_R \left[\begin{array}{l} \forall\delta_2 : AF_{in}\, p([w\gamma_1\gamma_2\gamma_5],[w\gamma_1\gamma_2\gamma_5\delta_2]) \wedge \\ \forall\delta_3 : AF_{in}\, q([w\gamma_1\gamma_2\gamma_5],[w\gamma_1\gamma_2\gamma_5\delta_3]) \end{array}\right] \end{array}\right]$$

The term $[w\gamma_1\gamma_2\gamma_3]$ denotes the world accessible from w via γ_1, γ_2, γ_3 transitions; it corresponds to the first sequence of three nested $\Diamond$-operators. The variable is replaced by the term $[w\gamma_1\gamma_2\delta_1]$; the latter denotes an element in the domain of the world $[w\gamma_1\gamma_2]$. The function δ_1 maps this world to the domain element. The quantifier $\forall\delta_1 : AF_{in}$, which originated from the $\forall x$-quantifier, ensures that all domain elements are covered.

Unfortunately, the quantifier exchange operator Υ which helped us to turn the functional translation into the optimized functional translation, cannot be used in the setting of quantified modal logic. An example (due to Andreas Herzig) shows what can happen. The formula $\Box(\exists x\,(p(x) \wedge \Box\Diamond\neg p(x)))$ is true at the world w_1 of the following model.

[12]The interpretations for quantified modal logics are restricted to non-empty domains. Therefore $\forall w\ \exists x\, in(w,x)$ can always be assumed.

[13]$FT(w,A)[x \mapsto [w\delta]]$ means translating A and then replacing all occurrences of x with $[w\delta]$. The translation rules for the domain quantifiers eliminate the *in* predicate as well. This need not be done if one wants to keep the domain variables as they are.

$$\begin{tikzcd}
 & w_2 \quad p(a), \neg p(b) \\
w_1 \arrow[ur, leftrightarrow] \arrow[dr, leftrightarrow] & \\
 & w_3 \quad p(b), \neg p(a).
\end{tikzcd}$$

For every world u accessible from w_1 (these are w_2 and w_3) and for some object x, $p(x)$ holds (for w_2, x is a and for w_3, x is b), and for every world v accessible from u (w_1) there is a world y accessible from v (w_2 and w_3 are the candidates) such that $\neg p(x)$ holds at y. Now we have to choose either w_2 or w_3 and check whether $\neg p(x)$ holds. However, x was determined in a previous world, in the case that $u = w_2$, x is a and in the case that $u = w_3$, x is b. Our choice depends on the path we took to get to $v = w_1$. This example shows that we must be careful where we apply the trick of moving existential quantifiers to the front. For quantified modal logic the trick does not work in general.

5.28. EXAMPLE. Next, we turn to intuitionistic logic. The "functional version" of the restriction on the assignment of predicates (4.3) that we find in intuitionistic logic is

$$\forall w\, \forall \gamma : AF_R\, (p(w) \to p([w\gamma])). \tag{5.8}$$

This axiom can easily be turned into a theory resolution rule.

$$\begin{array}{ll}
p(s), C & \\
\neg p([s't]), D & \sigma \text{ unifies } s \text{ and } s' \\
\hline
C\sigma, D\sigma &
\end{array}$$

5.29. EXAMPLE. Functional translations can always be tried when the binary relations occur in the typical guarded patterns $\forall y\, (R(x,y) \to \ldots)$ and $\exists y\, (R(x,y) \wedge \ldots)$. This is the case, for example, in (4.6), the standard translation for classical modal logics with neighborhood semantics. Here it is the neighborhood predicate N which can be replaced by functions AF_N, mapping worlds to their neighborhoods. Assuming seriality of N (each world has at least one neighborhood) one can define a functional translation FT as follows:

$$\begin{array}{rcl}
FT_c(w, \Box A) & = & \exists \gamma : AF_N\, \forall v\, (in([w\gamma], v) \leftrightarrow FT_c(v, A)) \\
FT_c(w, \Diamond A) & = & \forall \gamma : AF_N\, \exists v\, (in([w\gamma], v) \leftrightarrow FT_c(v, A)).
\end{array}$$

It is possible, although it would not make much sense, to replace the in-predicate with functions AF_{in}; in does not occur in the typical modal pattern. Therefore, the translation introduces equations and in-equations requiring equation handling in the theorem prover.

The standard translation for classical modal logic with the stronger semantics is

$$\begin{array}{rcl}
ST'_c(w, \Box A) & = & \exists V\, (N(w, V) \wedge \forall v\, (in(V, v) \to ST'_c(v, A))) \\
ST'_c(w, \Diamond A) & = & \forall V\, (N(w, V) \to \exists v\, (in(V, v) \wedge ST'_c(v, A))).
\end{array}$$

Now it does make sense to replace both relations, N and in, by accessibility functions. Assuming seriality of N and in, i.e., each world has a non-empty neighborhood, we get a much more compact functional translation:

$$\begin{array}{lcl} FT'_c(w, \Box A) & = & \exists\gamma : AF_N\, \forall\delta \in AF_{in}\, FT'_c([w\gamma\delta], A) \\ FT'_c(w, \Diamond A) & = & \forall\gamma : AF_N\, \exists\delta : AF_{in}\, FT'_c([w\gamma\delta], A). \end{array}$$

Intuitively, γ maps the world w to a neighborhood of w and δ maps this neighborhood back to a world.

5.30. EXAMPLE. It is instructive to see what happens if we try to prove the **K**-axiom $\Box(p \to q) \to (\Box p \to \Box q)$ using the above translation for classical modal logic. Since the **K**-axiom does not hold in classical modal logic, a resolution refutation should fail. The negation of the **K**-axiom is $\Box(p \to q) \wedge \Box p \wedge \Diamond\neg q$. The translation yields three clauses:

$\neg p([w_0 a\gamma]) \vee q([w_0 a\gamma])$	a is a Skolem constant
$p([w_0 b\gamma])$	b is a Skolem constant
$\neg q([w_0 \gamma c])$	c is a Skolem constant.

After one resolution between the first and last clause, giving $\neg p([w_0 ac])$, no further inference step is possible. The clause set cannot be refuted.

6. The semi-functional translation

The semi-functional translation combines the advantages of the relational and the functional translation while trying to avoid their respective disadvantages. One of the advantages of the relational translation is that the translation result is "natural" because it mirrors the Kripke-semantics of modal logics. The background theory to be added to the translation result is simple, and, in particular, does not introduce "unnecessary" new equations. Its disadvantage is that it produces large formulae which open huge search spaces for automated theorem provers. The functional translation, on the other hand, provides us with a very compact translation result. The price for this advantage is that we may have to cope with an additional equational theory. The introduction of theory-unification algorithms may simplify matters, though.

The semi-functional translation also produces a fairly compact translation result (although not as compact as the functional translation) and, like the relational translation, it does not introduce new equational theories. It even turns out that its syntactic peculiarities allow us to simplify background theories such that they can sometimes be reduced to simple unit clauses.

The semi-functional translation is called *semi-functional* because the operator $\Diamond$ is translated functionally whereas the $\Box$ is translated as in the relational approach. Just as with the functional translation its definition depends on whether we consider

$$\begin{array}{rcl} SF(w,p) & = & p(w) \\ SF(w,\neg A) & = & \neg SF(w,A) \\ SF(w,A\wedge B) & = & SF(w,A)\wedge SF(w,B) \\ SF(w,\Box A) & = & \forall v:W\,(R(w,v)\to SF(v,A)) \\ SF(w,\Diamond A) & = & \exists\gamma:AF_R\,SF([w\gamma],A) \end{array}$$

Table 7: The semi-functional translation

serial or non-serial accessibility relations. Here, we will focus on the serial case. For non-serial accessibility relations similar additions have to be made as for the functional approach (see Nonnengart [1992, 1993, 1995, 1996]).

6.1. DEFINITION. The *semi-functional translation* from modal logic into first-order logic is given by the clauses given in Table 7.

Obviously, the only difference between the functional and the semi-functional translation seems to be the treatment of $\Diamond$-formulae. However, this is not quite the only difference. We must ensure that $\Box$ and $\Diamond$ are dual operators, i.e., that for any formula A it holds that $\Box A \leftrightarrow \neg\Diamond\neg A$. In the functional translation this is obviously guaranteed. In the semi-functional translation the duality of the translations of $\Box$ and $\Diamond$ is not automatically given.

6.2. PROPOSITION (Nonnengart [1993, 1995, 1996]). *The duality principle $\Box A \leftrightarrow \neg\Diamond\neg A$ holds if and only if $\forall w:W\,\forall\gamma:AF_R\,R(w,[w\gamma])$ holds.*

In words: regardless of the serial modal logic that we are considering, its background theory must contain the above formula to ensure duality between the modal operators. This *minimal background theory* states that every world that is accessible via the accessibility functions is also accessible via the accessibility relation.

Recall that a formula is in *negation normal form* if it contains no implication or equivalence and all negations only occur immediately in front of atoms. Every formula can easily be transformed into an equivalent formula in negation normal form.

6.3. THEOREM ([Nonnengart 1995]). *Let A be a uni-modal formula in negation normal form. Then A is unsatisfiable (in the serial case) iff $SF(x,A)$ cannot be satisfied on any model which satisfies both $\forall w:W\,\forall\gamma:AF_R\,R(w,[w\gamma])$ and the properties induced by the specific axiom schemata for the modal logic under consideration.*

Therefore, the semi-functional translation behaves as desired. But what has been gained so far? Our aim was to define a translation that produces compact results and does not introduce new equations into the background theory of the modal logic under consideration. The latter is obviously fulfilled by the semi-functional translation as shown in Theorem 6.3. As for the former, it is easy to show that after transformation into clause normal form there is no difference in the number of clauses generated in the semi-functional and in the functional approach (in the semi-functional approach the clauses are bigger, though). However, there is a third invariant of this translation approach which turns out to be useful: given a formula A, the clause normal form of $SF(u, A)$ does not contain any positive R-literal. Thus, positive R-literals can occur only in the background theory of the modal logic under consideration. Below, we will take advantage of this fact.

6.1. Saturating background theories

The fact that no positive R-literals occur in semi-functional translation of modal formulae can be used by pre-computing everything that can possibly be derived from the background theory, i.e., this theory gets *saturated*. Such a saturation characterizes the modal logic and is thus independent of the theorem to be proved.

6.4. DEFINITION. We call a clause C *p-positive* (*p-negative*) if there is a positive (negative) occurrence of a p-literal in C. If, in addition, C is not p-negative (not p-positive) then we call C *pure-p-positive* (*pure-p-negative*).

6.5. DEFINITION *((Partial) Saturation).* Let p be a designated predicate symbol and let $\mathcal{C}$ be a set of p-positive clauses. The *saturation* of $\mathcal{C}$ with respect to p is defined to be the set $\{C \mid \mathcal{C} \vdash_{\text{res}} C$ and C is pure-p-positive$\}$, i.e., the set of clauses we obtain by resolution within $\mathcal{C}$, and whose elements are pure-p-positive.

As an example, consider the clause set $\{p(a), \neg p(x) \vee p(f(x))\}$. The only clause which is pure-p-positive is $p(a)$. But it is possible to derive more pure-p-positive clauses by resolution, namely all atoms of the form $p(f^n(a))$, for $n \geq 0$.

Knowing about the saturation of a given clause set is often quite helpful.

6.6. LEMMA. *Let $\mathcal{C}$ be a clause set and let p be a designated predicate symbol. Let $\mathcal{C}' \subseteq \mathcal{C}$ be the subset of $\mathcal{C}$ whose elements are positive with respect to p. If $\mathcal{C}''$ is the saturation of $\mathcal{C}'$ with respect to p then $\mathcal{C}$ is unsatisfiable iff $(\mathcal{C} \setminus \mathcal{C}') \cup \mathcal{C}''$ is unsatisfiable.*

The problem with the above lemma is that saturations are usually infinite. However, if we are able to find a finite alternative clause set with exactly the same saturation we can use this one instead.

6.7. THEOREM. *Let $\mathcal{C}$ be a finite clause set, let p be a designated predicate symbol, and let $\mathcal{D} \subseteq \mathcal{C}$ contain the p-positive clauses of $\mathcal{C}$. Moreover, let $\mathcal{B}$ be a finite set of p-positive clauses whose saturation with respect to p is identical to $\mathcal{D}$'s saturation with respect to p. Then $\mathcal{C}$ is unsatisfiable iff $(\mathcal{C} \setminus \mathcal{D}) \cup \mathcal{B}$ is unsatisfiable.*

Thus, the idea is to extract $\mathcal{D}$ and to find a simpler clause set $\mathcal{B}$ with the same saturation.

6.8. Example. Consider the simple background theory given by the clauses:

$$\begin{array}{l} p(a) \\ p(f(a)) \\ \neg p(x) \vee \neg p(f(x)) \vee p(f(f(x))) \end{array} \tag{6.1}$$

Its saturation is $\{p(f^n(a)) \mid n \geq 0\}$. One may prove this is as follows. First, each of these elements can indeed be derived. A simple induction on n will do. For the induction step assume that $p(f^k(a))$ is derivable for all $k < n$. So both $p(f^{n-2}(a))$ and $p(f^{n-1}(a))$ are derivable and therefore $p(f^n(a))$ can be obtained by two resolution steps with the third clause from the original clause set. Thus $\{p(f^n(a)) \mid n \geq 0\}$ is at least contained in the saturation we are looking for.

It remains to show that the saturation is contained in the derived clause set. To this end we show that any pure p-positive clause which is derivable from $\{p(f^n(a)) \mid n \geq 0\}$ and the p-positive clauses of the clause set under consideration is already of the form $p(f^n(a))$. Evidently, resolution steps between $p(f^k(a))$, $p(f^l(a))$ and $\neg p(x) \vee \neg p(f(x)) \vee p(f(f(x)))$ are possible only if $k = l + 1$ (or $l = k + 1$) and they result in $p(f^{l+2}(a))$ ($p(f^{k+2}(a))$ respectively). This derived unit clause does indeed belong to $\{p(f^n(a)) \mid n \geq 0\}$ and we are done.

Now consider the somewhat "simpler" clause set

$$\begin{array}{l} p(a) \\ \neg p(x) \vee p(f(x)). \end{array} \tag{6.2}$$

The saturation of this clause set is also $\{p(f^n(a)) \mid n \geq 0\}$ and therefore (Theorem 6.7) this new clause set may be used to replace the original background theory.

Observe that the two clause sets (6.1) and (6.2) are not equivalent. It is the mere fact that they form a background theory in the sense that they contain the only p-positive literals occurring anywhere in the clause set being considered which allows us to perform such a "simplification." Hence, what we use here is not just that the background theory is *something* we know about p but is indeed *all* we know about p.

For two well-known serial modal logics the saturation approach does not lead to anything new, namely **KD** and **KT**. However, the background theories for these modal logics are represented by one or two unit clauses anyway, and, thus, the proof search will not be influenced too heavily.[14]

6.9. Example. Consider the logic **KDB**, which is axiomatized by the additional axiom schemata $\Box p \to \Diamond p$ and $p \to \Box\Diamond p$. These two schemata characterize seriality

[14]Note that for the logic **KD** we could incorporate this very unit clause directly in the translation description. Interestingly, this would result in exactly the same clause set we would get if we applied the functional translation approach.

and symmetry of the underlying accessibility relation; see Table 2. Therefore, the background theory for **KDB** is:

$$\begin{aligned}&\forall w : W\, \forall\gamma : AF_R\, R(w, [w\gamma])\\ &\forall u, v : W\, (R(u, v) \to R(v, u)).\end{aligned} \tag{6.3}$$

The saturation of this background theory can easily be found, and we end up with

$$\begin{aligned}&\forall w : W\, \forall\gamma \in AF_R\, R(w, [w\gamma])\\ &\forall w : W\, \forall\gamma : AF_R\, R([w\gamma], w).\end{aligned}$$

Hence, these two unit clauses are sufficient as the background theory for **KDB**. Although this only seems to be a minor improvement over (6.3), such a replacement at least avoids undesirable cycles in the search space.

6.10. Example. Consider the modal logic **S4**, which is characterized by reflexivity and transitivity (see Table 2); the corresponding axiom schemata are $\Box p \to p$ and $\Box p \to \Box\Box p$. The background theory is given by

$$\begin{aligned}&\forall w : W\, \forall\gamma : AF_R\, R(w, [w\gamma])\\ &\forall w : W\, R(w, w)\\ &\forall u, v, w : W\, (R(u, v) \wedge R(v, w) \to R(u, w)).\end{aligned} \tag{6.4}$$

Again, we have to saturate this clause set bearing in mind that this is indeed all we know about R, for any formula to be proved unsatisfiable (in **S4**) will not contain R-positive clauses. Let us show now that the saturation consists of the (infinite) set of unit clauses of the form $\{R(w, [w\gamma_1 \ldots \gamma_n]) \mid n \geq 0\}$.

To this end we show that the purely positive R-clauses in the background theory (which are $R(w, [w\gamma])$ and $R(w, w)$) are contained in this set; this is trivial. Then we have to show that resolving upon two arbitrary elements of the candidate saturation of the transitivity clause does not produce anything new, and indeed, resolving $R(w, [w\gamma_1 \ldots \gamma_n])$ and $R(w, [w\delta_1 \ldots \delta_m])$ with the first two literals in $R(u, v) \wedge R(v, w) \to R(u, w)$ results in $R(w, [w\gamma_1 \ldots \gamma_n\delta_1 \ldots \delta_m])$ which is already contained in $\{R(w, [w\gamma_1 \ldots \gamma_n]) \mid n \geq 0\}$. Finally, we have to show that the candidate saturation is not too large, i.e., that each of its elements can in fact be derived and this follows by a simple induction on n.

At this stage we have found the saturation of the **S4** background theory. Next, we have to find an alternative clause set with the same saturation but which is somehow simpler than the original one. Finding such an alternative is still to be performed by a *good guess*; it is not yet known how this can be automated. For the present example it is not hard to find a suitable clause set, namely

$$\begin{aligned}&\forall u : W\, R(u, u)\\ &\forall u, v : W\, \forall\gamma : AF_R\, (R(u, v) \to R(u, [v\gamma]))\end{aligned} \tag{6.5}$$

or, equally simple,

$$\begin{array}{l}\forall u : W\, R(u,u)\\ \forall u,v : W\, \forall\gamma : AF_R\, (R([u\gamma],v) \to R(u,v)).\end{array}$$

It is easy to show that the saturation of this clause set is identical to the saturation of the **S4** background theory, and what we have gained is that we may replace the background theory for **S4** by the two simpler clauses in (6.5). In particular, the deletion of the transitivity clause turns out to be of major importance.

Our next example illustrates the effect of the semi-functional translation together with saturation with a little example.

6.11. Example. We want to prove the validity of the formula $\Diamond\Box p \leftrightarrow \Diamond\Box\Diamond\Box p$ in **S4**. After negating and translating semi-functionally we end up with the following clause set (where γ, a, b are Skolem constants, f, h, k are Skolem functions and all variables are assumed to be universally quantified):

$$\begin{array}{l}\neg R([\gamma a],u) \vee \neg R([\gamma b],v) \vee \neg R([vf(v)],w) \vee p(u) \vee p(w)\\ \neg R(\gamma,u) \vee \neg p([u.g(u)]) \vee \neg R(\gamma,v) \vee \neg R([vh(v)],w) \vee \neg p([wk(w,v)])\\ R(u,u)\\ \neg R(u,v) \vee R(u,[v\gamma_1]).\end{array}$$

This clause set is much smaller than what we would obtain from the relational translation; moreover, the search space has been reduced to such an extent that no standard FO theorem prover will have difficulties with it.

It should now be obvious how a logic like **KD4** has to be treated. Its background theory is described by

$$\begin{array}{l}\forall u : W\, \forall\gamma : AF_R\, R(u,[u\gamma])\\ \forall u,v,w : W\, ((R(u,v) \wedge R(v,w)) \to R(u,w)),\end{array}$$

and it can easily be shown that the saturation of this theory is

$$\{R(u,[u\gamma_1 \ldots \gamma_n]) \mid n \geq 1, \gamma_i : AF_R\},$$

i.e., it differs from the saturation of the **S4**-theory only in that it lacks the reflexivity clause $R(u,u)$. An alternative clause set for this background theory, with the same saturation, is easily found:

$$\begin{array}{l}\forall u : W\, \forall\gamma \in AF_R\, R(u,[u\gamma])\\ \forall u,v : W\, \forall\gamma : AF_R\, (R(u,v) \to R(u,[v\gamma])).\end{array}$$

So far only a few modal logics have been examined with respect to semi-functional translation and saturation of background theories. There is a property which allows us to broaden the scope of the method, viz. the so-called rootedness property of modal frames. Let us first have a look at the technique applied to the logic **S5**.

6.12. EXAMPLE. **S5** can be axiomatized by the schemata $\Box p \to p$, $\Box p \to \Box\Box p$, and $p \to \Box\Diamond p$ or, equivalently, by $\Box p \to p$ and $\Diamond p \to \Box\Diamond p$. The corresponding properties of the accessibility relation are reflexivity, transitivity and symmetry in the former and reflexivity and euclideanness in the latter axiomatization; the two are obviously equivalent. Now, the saturation of either background theory consists of all unit clauses of the form $\{R([u\gamma_1 \dots \gamma_n], [u\delta_1 \dots \delta_m]) \mid n, m \geq 0\}$. Also, it is easy to find an alternative clause set which is simpler than the original one, but generates the same saturation. The axioms are

$$\begin{aligned}
&\forall u : W\, R(u, u)\\
&\forall u, v : W\, \forall \gamma : AF_R\, (R(u, v) \to R(u, [v\gamma]))\\
&\forall u, v : W\, \forall \gamma : AF_R\, (R(u, v) \to R([u\gamma], v)),
\end{aligned}$$

which are still rather complicated. This background theory can be significantly simplified if we exploit the fact that we need only consider *rooted* frames as defined in [Segerberg 1971].

6.13. DEFINITION *(Rooted Frames).* A frame $\mathcal{F} = (W, R)$ is called *rooted* if there exists a world w in W such that for every world v in W it holds that $R^*(w, v)$, where R^* denotes the reflexive and transitive closure of R.

Hence, in a rooted frame any world can be reached from an initial world by zero or more R-steps and it is thus impossible to have two unconnected "islands" of worlds.

6.14. DEFINITION *(Generated Frames).* Given a frame $\mathcal{F} = (W, R)$ and an arbitrary world w in W we define $W' = \{v \in W \mid R^*(w, v)\}$ and $R' = R \cap (W' \times W')$. The frame (W', R') is called the frame *generated from* $\mathcal{F}$ (with initial world w).

Evidently, every generated frame is rooted. Modal logics are not able to distinguish between rooted and non-rooted frames and this is shown by the following result.

6.15. LEMMA (Segerberg). *Let $\mathcal{F}$ be a modal logic frame and let $\Im = (\mathcal{F}, P)$ be an arbitrary interpretation based on $\mathcal{F}$. Let $\Im' = (\mathcal{F}', P)$ where $\mathcal{F}'$ is the frame generated from $\mathcal{F}$. Then, for all worlds $v \in W'$ and for all modal formulae A,*

$$\Im, v \models A \text{ iff } \Im', v \models A.$$

PROOF. The identity relation is a bisimulation between $\Im$ and $\Im'$, as the reader can easily check. Hence, the lemma follows by Proposition 4.16. $\Box$

As a consequence of the lemma, we may restrict our attention to rooted frames. What does rootedness actually mean in (semi-)functional frames?

6.16. LEMMA. *A frame (interpretation) is rooted (with initial world w) iff for every world u there exist some $\gamma_1, \dots, \gamma_n \in AF_R$ ($n \geq 0$) such that $u = [w\gamma_1\gamma_2\gamma_3 \dots \gamma_n]$.*

Proof. In rooted frames each world u can be reached from the initial world w by a finite sequence of R-transitions, i.e., there exist worlds $w_1, \dots, w_{n-1}$ such that $R(w, w_1)$, $R(w_1, w_2)$, ..., $R(w_{n-1}, u)$. In the extended functional frames this is just $[w\gamma_1 \dots \gamma_n]$ for suitable γ_i. $\square$

We may therefore assume that the property $\forall u \exists \gamma_1, \dots, \gamma_n\, (u = [w_0\gamma_1 \dots \gamma_n])$ holds, where w_0 is the initial world. This can be used for further optimizations of the semi-functional translation.

6.17. Example. Recall from Example 6.12 that the saturation of **S5** resulted in an infinite clause set consisting of unit clauses of the form $R([u\gamma_1 \dots \gamma_n], [u\delta_1 \dots \delta_m])$ with $n, m \geq 0$ (and universally quantified variables u, γ_i, δ_j). Now consider the subset we get after instantiating the variable u with w_0, the initial world. Then both arguments of the R-literals are of the form $w_0\gamma_1 \dots \gamma_n$, where every γ_i is universally quantified, and therefore this term can represent any world. Thus, we know that (given the rootedness assumption) $R([w_0\gamma_1 \dots \gamma_n], [w_0\delta_1 \dots \delta_m])$ can be simplified to $R(v, w)$, i.e., the universal relation, which obviously subsumes all of the unit clauses described by $R([u\gamma_1 \dots \gamma_n], [u\delta_1 \dots \delta_m])$. More formally:

$$\begin{aligned}&\forall u\, \exists \gamma_1, \dots, \gamma_n\, (u = [w_0\gamma_1 \dots \gamma_n]) \Rightarrow \\ &\qquad \forall u, \gamma_i, \delta_j\, R([u\gamma_1 \dots \gamma_n], [u\delta_1 \dots \delta_m]) \leftrightarrow \forall v, w\, R(v, w).\end{aligned}$$

Thus, instead of considering the still rather complicated background theory for **S5** as described above, we can simplify it.

6.18. Example. Consider **KD5** and **KD45**. These are axiomatized by the clause set

$$\begin{aligned}&\forall u : W\, \forall \gamma : AF_R\, R(u, [u\gamma]) \\ &\forall u, v, w : W\, (R(u, v) \wedge R(u, w) \to R(v, w))\end{aligned} \qquad (\mathbf{KD5})$$

and

$$\begin{aligned}&\forall u : W\, \forall \gamma : AF_R\, R(u, [u\gamma]) \\ &\forall u, v, w : W\, (R(u, v) \wedge R(v, w) \to R(u, w)) \\ &\forall u, v, w : W\, (R(u, v) \wedge R(u, w) \to R(v, w))\end{aligned} \qquad (\mathbf{KD45})$$

respectively, i.e. seriality and euclideanness (for **KD5**) and, additionally, transitivity (for **KD45**). Their saturations consist of the unit clause sets with all elements of the form $R([u\gamma_1 \dots \gamma_n], [u\delta_1 \dots \delta_m])$, where $m, n \geq 1$ for **KD5**, and $m \geq 1$ and $n \geq 0$ for **KD45**; in addition, the saturation of **KD5** also contains $R(u, [u\gamma])$. Both are quite similar to **S5**.

Unfortunately, since $m \geq 1$ (and $n \geq 1$ for **KD5**) these two arguments are not yet in the form that the rootedness assumption can be applied. However, since $m \geq 1$ we know that $m - 1 \geq 0$ and we therefore get for all $m \geq 1$ and $n \geq 0$, i.e., for **KD45**

$$\begin{aligned}&\forall u\, \exists \gamma_1, \dots, \gamma_k\, (u = [w_0\gamma_1 \dots \gamma_k]) \Rightarrow \\ &\qquad \forall u, \gamma_i, \delta_j\, R([u\gamma_1 \dots \gamma_n], [u\delta_1 \dots \delta_m]) \leftrightarrow \forall v, w, \gamma\, R(v, [w\gamma])\end{aligned} \qquad (6.6)$$

and for all $m \geq 1$ and $n \geq 1$, and so for **KD5**

$$\forall u\, \exists \gamma_1, \ldots, \gamma_k\, (u = w_0 \gamma_1 \ldots \gamma_k) \Rightarrow \\ \forall u, \gamma_i, \delta_j\, R([u\gamma_1 \ldots \gamma_n], [u\delta_1 \ldots \delta_m]) \leftrightarrow \forall v, w, \gamma, \delta\, R([v\gamma], [w\delta]). \quad (6.7)$$

The unit clause $R(v, [w\gamma])$ that we got for **KD45** subsumes the whole saturation and can therefore be used as the background theory for the logic **KD45**, whereas the unit clause $R([v\gamma], [w\delta])$ subsumes almost the whole saturation for **KD5**; the only clause which is not subsumed is $R(u, [u\gamma])$ and therefore the background theory for **KD5** can be described by the two unit clauses $R(u, [u\gamma])$ and $R([v\gamma], [w\delta])$.[15]

Table 8 summarizes the background theories for well-known serial modal logics in the setting of the semi-functional translation; all variables are universally quantified.

logic	background theory
KD	$R(u, [u\gamma])$
KT	$R(u, [u\gamma])$ and $R(u, u)$
KDB	$R(u, [u\gamma])$ and $R([u\gamma], u)$
KD4	$R(u, [u\gamma])$ and $(R(u, v) \rightarrow R(u, [v\gamma]))$
S4	$R(u, u)$ and $R(u, v) \rightarrow R(u, [v\gamma])$
KD5	$R(u, [u\gamma])$ and $R([u\gamma], [v\delta])$
KD45	$R(u, [v\gamma])$
S5	$R(u, v)$

Table 8: Logics and their background theories

How should these results, and in particular those for the logics **S5**, **KD45**, and **KD5**, be interpreted? Their simplicity and generality may be surprising. As Segerberg found out by examining the model theory for various modal logics [Segerberg 1971], the characteristic frames for **S5** are so-called *clusters*; that is, sets of worlds such that each world has access to any other world including itself. Thus we may assume that the accessibility relation for **S5** is the universal relation over $W \times W$.

The models for **KD45** are characterized by either a single cluster (as for **S5**) or a single world together with a cluster such that this particular world has access to each element of the cluster. The characteristic frames for **KD5** differ from those for **KD45** in that the single world does not necessarily have access to *all* elements in the cluster.

If we compare Segerberg's results with the saturated background theory we obtained for these logics, we see that for **S5** we do indeed find the universal accessibility relation; that for **KD45** every world does indeed have access to every other

[15]Actually, this can be even further simplified, for this first unit clause is subsumed for every instance but one, namely $\forall \gamma\, R(w_0, [w_0\gamma])$, where w_0 denotes the initial world. Hence, the two unit clauses $\forall \gamma\, R(w_0, [w_0\gamma])$ and $\forall v, w, \gamma, \delta\, R([v\gamma], [w\delta])$ would already suffice in case of **KD5**.

world which itself is somehow accessed; and that for **KD5** any two worlds can indeed access each other provided both have predecessors. It is remarkable how Segerberg's model-theoretic results are mirrored in the saturation approach and that with essentially syntactic means; the extent of the correspondence remains to be explored.

6.2. From saturations to inference rules

For many modal logics the semi-functional translation allows us to simplify the background theory after saturation to a few unit clauses. There are exceptions, however, for which the saturation is not that successful, although the simplifications are still significant.

The idea we will pursue in this section is to cast the whole saturation set into a suitable inference rule instead of trying to find an alternative clause set for the saturation found. We will illustrate how this can be done for **S4**. Recall that the saturation for **S4** is

$$\begin{array}{l} R(u,u) \\ R(u,[u\gamma]) \\ \ldots \\ R(u,[u\gamma_1\ldots\gamma_n]) \\ \ldots \end{array}$$

Observe that each first argument of the respective R-literals is a variable and that each second argument "starts" with the same variable. This observation guarantees that — given an arbitrary unsatisfiable formula to be refuted — a corresponding finite and unsatisfiable set of ground instances of clauses taken from this formula contains negative R-literals only of the form $\neg R(s,[s\alpha_1\ldots\alpha_k])$. It suffices to unify the first argument of such a negative R-literal with a *prefix* of its second argument and thus to forget about the background theory or its saturation.

6.19. DEFINITION *(The* **S4** *Inference Rule).* Let s and $t.\alpha_1\ldots\alpha_n$ be two world terms and let C be a clause. Then the **S4** *inference rule* is the following rule

$$\frac{\neg R(s,[t\alpha_1\ldots\alpha_n])\vee C}{\sigma C} \qquad n\geq 0,$$

where σ is the most general unifier of (s,t).

The above inference rule is the only inference mechanism that has to be added to a clause-based theorem prover in order to obtain refutation completeness for **S4**.

6.20. DEFINITION *(The* **S4** *Inference System).* The **S4** *inference system* consists of the standard resolution and factorization rules together with the **S4** inference rule.

6.21. THEOREM. *Let A be a uni-modal formula in negation normal form. Then A is* **S4**-*unsatisfiable iff the* **S4** *inference system refutes* $\exists w\, SF(w, A)$.

Similar inference systems can be defined for other (more complicated) modal and temporal logics; see Nonnengart [1993, 1995, 1996]. Since the functional translation for modal logics above **KD** require special unification algorithms, and the optimized semi-functional translation requires special inference rules, both require modifications of theorem provers. This has not yet been implemented, and therefore there are no systematic performance comparisons available.

7. Variations and alternatives

In this section we briefly discuss a number of alternative encodings of nonclassical logics into first-order logic and into other logics. We begin by mixing syntactic and semantic encodings. After that we discuss a number of internal, 'modal-to-modal' translations that will allow us to use FO tools to reason in non-first-order definable modal logics. We conclude the section with two flavors of encoding that escape the first-order realm but that have proved to be important for working with a large number of logics.

7.1. Mixing Hilbert-style and semantic reasoning

The translation approaches presented so far depend on the semantics of the logic under consideration, as the translation rules for the connectives and operators are derived from their semantics. In addition, first-order axiomatizations of the semantic structures are needed. If the semantics is not clear, or not first-order, the translation approaches cannot work in the usual way. But, as long as (parts of) the logic have a suitable Hilbert axiomatization, it is possible to combine Hilbert-style reasoning with the translation method [Ohlbach 1998*a*]. To this end the Hilbert system is encoded in first-order logic, not exactly as shown in Section 3, but similarly. Instead of the unary truth predicate T, we use a binary truth predicate $T(A, m)$. Intuitively it means "A is true in the model m."

7.1. DEFINITION *(T-Encoding).* A Hilbert axiom A with the predicate variables p_1, ..., p_n is *T-encoded* as

$$\forall p_1, \dots, p_n\, \forall m\, T(A, m).$$

An inference rule "from $\vdash A_1, \dots, \vdash A_k$ infer $\vdash A$" with the predicate variables $p_1, \dots, p_n$ is *T-encoded* as

$$\forall p_1, \dots, p_n\, (\forall m\, T(A_1, m) \wedge \cdots \wedge \forall m\, T(A_k, m) \rightarrow \forall m\, T(A, m)).$$

If A is a Hilbert axiom or rule then let $\tau(A)$ be its T-encoding.

The second argument of the T-predicate is completely redundant for the Hilbert system itself. It becomes important in combination with the second part of the specification, the "T-encoding" of the semantics of those operators for which a semantics is known. In the combined method it is not necessary to have a semantics and a translation rule for all operators. If, however, there is a suitable semantics for a given operator, it is encoded as a predicate logical equivalence which can be used immediately as a translation rule.

For example the semantics of the classical connective $\wedge$ (conjunction) is

$$m \models A \wedge B \text{ iff } m \models A \,\&\, m \models B.$$

Using a binary predicate $T(A, m)$ this can be expressed as a FO equivalence

$$\forall A, B\, \forall m\, (T(A \wedge B, m) \;\leftrightarrow\; (T(A, m) \wedge T(B, m))). \tag{7.1}$$

The semantics of the other classical connectives can be expressed in a similar way; observe that the symbols for the Boolean connectives are at the formula level and at the term level.

Examples for "T-encoded" semantics of nonclassical operators are:

$$\begin{aligned}
\forall A\ \forall m\ (T(\Box A, m) &\;\leftrightarrow\; \forall m'\ R(m, m') \to T(A, m')) && (7.2)\\
\forall A\ \forall m\ (T(\Diamond A, m) &\;\leftrightarrow\; \exists m'\ R(m, m') \wedge T(A, m')) && (7.3)\\
\forall A\ \forall m\ (T(\Box A, m) &\;\leftrightarrow\; \forall \gamma\ T(A, [m\gamma])) && (7.4)\\
\forall A\ \forall m\ (T(\Diamond A, m) &\;\leftrightarrow\; \exists \gamma\ T(A, [m\gamma])) && (7.5)\\
\forall A\ \forall m\ (T(\bigcirc A, m) &\;\leftrightarrow\; T(A, m+1)) && (7.6)\\
\forall A, B\, \forall m\, (T(A \to B, m) &\;\leftrightarrow\; \forall m'\, (R(m, m') \to T(A, m') \to T(B, m'))) && (7.7)
\end{aligned}$$

Equations (7.2) and (7.3) represent the standard possible worlds semantics for the modal operators $\Box$ and $\Diamond$ for modal logics above **K**; (7.4) and (7.5) describe the corresponding "functional" semantics for modal logics above **KD**. The γ are accessibility functions, (7.6) represents the semantics of the temporal *next* operator $\bigcirc$ in an integer-like time structure, and (7.7) is the semantics of intuitionistic implication.

The *T-encoded semantics* for the corresponding connectives and operators can be used to rewrite (translate) a term-encoded formula to the formula level of predicate logic. For example, $T(\Box(A \wedge B), m)$ can be rewritten to

$$\forall m'\, (R(m, m') \to T(A \wedge B, m'))$$

using (7.2) and then, using (7.1), further to

$$\forall m'\, (R(m, m') \to (T(A, m') \wedge T(B, m'))).$$

7.2. Definition *(Semantic Normalizing).* Given a set S of T-encoded semantics for some operators, let $\pi_S(T(A, m))$ be the formula obtained by applying the equivalences in S to $T(A, m)$ exhaustively from left to right. We call $\pi_S(T(A, m))$ the *semantically normalized* or *S-normalized* formula.

Semantic normalizing is an equivalence preserving transformation. In other words,

$$(S \wedge T(A, m)) \leftrightarrow (S \wedge \pi_S(T(A, m)))$$

holds universally. If S represents the semantics of *all* the connectives and operators occurring in some (ground) formula A, then $\forall m\ \pi_S(T(A, m))$ is essentially a FO-translated nonclassical formula. The only difference to the translation methods presented above is that in the final result of the translation, literals $T(p, m)$ are usually replaced with $p(m)$. Completeness of the semantics means that after the rewrite step the equivalences are not needed any more.

If p is a constant, there is no big difference between $T(p, m)$ and $p(m)$. If p, however, is a predicate variable, then there is a big difference between $T(p, m)$ and $p(m)$. The first literal is first-order, whereas the second literal is second-order. For clause sets of the first kind, the Herbrand Theorem applies. Whenever there is a refutation proof for some theorem, then there is also a ground refutation proof, where the variable p in $T(p, m)$ is instantiated with a term-encoded formula. This cannot be guaranteed for literals $p(m)$ with predicate variables p. Therefore, in this case $p(m)$ is stronger than $T(p, m)$. Since predicate variables implicitly quantify over formulae in a Hilbert calculus, the first-order version $T(p, m)$ is appropriate, but the second-order version $p(m)$ is not.

The set S need not contain T-encoded semantics for *all* the connectives. For example, if S only contains the semantics of the classical connectives, then

$$\pi_S(T(\Box(p \wedge q) \vee r, m)) = T(\Box(p \wedge q), m) \vee T(r, m).$$

The "$\wedge$" inside the $\Box$ cannot be rewritten at this step, but it might be rewritten after some inference steps which bring the $\wedge$ to the top-level of the term.

Semantic normalizing can simplify T-encoded Hilbert axioms. For example the **K**-axiom "$\Box A \wedge \Box(A \to B) \to \Box B$" for modal logic is T-encoded as

$$\forall A, B\ \forall m\, (T((\Box A \wedge \Box(A \to B)) \to \Box B, m)) \tag{7.8}$$

and S-normalized without using a semantics of the $\Box$-operator to

$$\forall A, B\ \forall m\, (T(\Box A, m) \wedge T(\Box(A \to B), m) \to T(\Box B, m)). \tag{7.9}$$

The necessitation rule "from $\vdash A$ derive $\vdash \Box A$" is T-encoded as

$$\forall A\, (\forall m\, T(A, m) \to \forall m\, T(\Box A, m)). \tag{7.10}$$

7.3. DEFINITION *(Mixed Problem Specification).* A mixed problem specification $(S,\ \tau_{\mathcal{F}}, C, \neg\, Th)$ consists of a set S of T-encoded semantics, a set $\tau_{\mathcal{F}}$ of axioms for restricting the semantic structures, a set C of T-encoded Hilbert axioms and rules and a *negated* T-encoded candidate theorem of the form $\exists\overline{p}\forall\overline{q}\exists m\, \neg T(A, m)$. It represents the problem of proving $\forall\overline{p}\exists\overline{q}\forall m\ T(A, m)$ in the logic specified by $S, \tau_{\mathcal{F}}$ and C.

The problem specification is *S-normalized* if all atoms $T(A, m)$ in $\tau_{\mathcal{F}}, C, \neg\, Th$ are S-normalized. The connectives with a semantics definition in S are the *defined connectives.* All other connectives are the *undefined* or *axiomatized* connectives.

The above schema is not the most general one. In some logics, for example in relevance logic, a theorem is supposed to hold only in some selected world 0. In this case one must refute the formula $\exists\overline{p}\forall\overline{q}\neg T(A,0)$ instead of the formula with the $\exists m$ quantification. For most logics it is obvious how to adapt this schema.

Equations (7.1)–(7.7) are examples for S. (7.8) and (7.10) are examples for C. $\tau_{\mathcal{F}}$ may, for example, contain the reflexivity axiom for the accessibility relation R used in (7.2). It may also contain the restrictions on the assignment which is necessary for intuitionistic logic: for all propositional constants p: $\forall m, m'\ ((T(p,m) \wedge R(m,m')) \rightarrow T(p,m'))$. The formulae in $\tau_{\mathcal{F}}$ may actually be represented by theory unification algorithms, or by theory resolution or constraint rules; see [Nonnengart 1995].

A T-encoded mixed problem specification is first-order and can be given to any FO theorem prover. Let us illustrate the combination of S-normalizing and resolution with a simple example from modal logic.

7.4. EXAMPLE. Suppose we want to prove $\Box(A \wedge B) \rightarrow \Box A$ from the **K**-axiom and the necessitation rule, and we want to use only the semantics of the classical connectives. This means that the S-part of the mixed problem specification $(S, \tau_{\mathcal{F}}, C, \neg Th)$ consists of clauses for the booleans only; $\tau_{\mathcal{F}}$ is empty, and C contains the **K**-axiom. The S-normal form of the T-encoded K-axiom (7.8) is

$$\neg T(\Box A, w) \vee \neg T(\Box(A \rightarrow B), w) \vee T(\Box B, w). \tag{7.11}$$

The clause form of the necessitation rule (7.10) is

$$\neg T(A, f(A)) \vee T(\Box A, w) \tag{7.12}$$

where f is a Skolem function. The negation of the T-encoded theorem $\Box(A \wedge B) \rightarrow \Box A$ is rewritten to

$$T(\Box(a \wedge b), w_0) \tag{7.13}$$

$$\neg T(\Box a, w_0) \tag{7.14}$$

where a, b and w_0 are Skolem constants. Two resolution steps with (7.11), (7.13) and (7.14) yield

$$\neg T(\Box(a \wedge b \rightarrow a), w_0).$$

This is resolved with the T-encoded necessitation rule (7.12). The resolvent $\neg T(a \wedge b \rightarrow a, f(a \wedge b \rightarrow a))$ is rewritten to the following refutable set:

$$T(a, f(a \wedge b \rightarrow a))$$
$$T(b, f(a \wedge b \rightarrow a))$$
$$\neg T(a, f(a \wedge b \rightarrow a)).$$

T-encoded Hilbert axioms are particularly suited for logics with a semantics which is appropriate for translation into predicate logic, but where the class of semantic structures is not first-order axiomatizable. The second-order properties of the semantics usually correspond to particular Hilbert axioms.

With some key examples, we show that it is possible to use the basic semantics for translation, whereas the critical Hilbert axioms are just T-encoded, and not turned into conditions on the semantic structures, which may be second-order. This simplifies the proof procedures considerably. Moreover, we do not get the completeness problems related to the transition from implicit quantifiers over formulae in the Hilbert axiom to second-order quantifiers over predicates in the standard translation [Sahlqvist 1975].

7.5. EXAMPLE. We show that the McKinsey axiom (see Example 5.19) together with the transitivity of the accessibility relation implies $\Diamond(p \to \Box p)$ (atomicity). Van Benthem's semantic proof of this theorem uses the axiom of choice ("it is as serious as this" [van Benthem 1984]). We use the **KD4** possible-worlds semantics (with seriality and transitivity of the accessibility) for the modal operators instead, and leave the McKinsey axiom essentially as it is. To make the example small enough we make use of the functional semantics $S = \{(7.4),(7.5)\}$.

The T-encoded and S-normalized McKinsey axiom, using (7.4) and (7.5) for the modal operators, is

$$\forall p\, \forall w\ (\exists a\, \forall x\, \neg T(p, [wax]) \vee \exists b\, \forall y\, T(p, [wby]))\,,$$

and so we obtain its clause form as $\neg T(p, [wa(w,p)x]) \vee T(p, [wb(w,p)y])$. The negated theorem $\Box(q \wedge \Diamond\neg q)$ is T-encoded and S-normalized to $T(q, [w_0u])$ and $\neg T(q, [w_0vc(v)])$. The empty clause is derivable in two resolution steps using the (transitive) unifier $\{p \mapsto q, w \mapsto w_0, u \mapsto a(w_0,q)x, v \mapsto b(w_0,q), y \mapsto c(b(w_0,q))\}$.

7.6. EXAMPLE. This example is from temporal logic. To make it more interesting, we choose an integer-like time structure such that an induction axiom holds:

$$p \wedge \Box(p \to \bigcirc p) \to \Box p. \tag{7.15}$$

Here, the $\Box$-operator means "always in the future" and the $\bigcirc$-operator means "at the next moment in time." The induction axiom (7.15) expresses: "if p holds now, and at all times t in the future, if p holds at time t then it holds at time $t+1$, then p will always hold in the future." The temporal semantics of the $\Box$- and $\bigcirc$-operators are functionally T-encoded as

$$\forall p\ \forall m\,(T(\Box p, m) \quad \leftrightarrow \quad \forall m'\ T(p, [mm'])) \tag{7.16}$$

$$\forall p\ \forall m\,(T(\bigcirc p, m) \quad \leftrightarrow \quad T(p, [m1])). \tag{7.17}$$

As an aside, the "functional" reading of an atom like $T(p, [mm_1 \dots m_n])$ is that p holds at a time point determined by starting at time point m, applying the function m_1 to m to get to time point $m_1(m)$ and so on. The *next time* operator $\bigcirc$ generates a constant 1, which is to be interpreted as the successor function. For example $T(p, [ma1])$ expresses that p holds at time point $a(m) + 1$. Notice that $[m1a1]$ and $[ma11]$ as the second argument of the T-predicate denote different time points, because $a(m+1)+1$ may be different from $a(m)+1+1$. Terms like $[ma11]$ may be abbreviated as $[ma2]$.

The T-encoded induction axiom is

$$\forall p\ \forall m\ (T(p,m) \wedge \forall n\,(T(p,[mn]) \rightarrow T(p,[mn1])) \rightarrow \forall n\, T(p,[mn])) \qquad (7.18)$$

and so we get as its clause form

$$\neg T(p,m) \vee T(p,[mf(p)]) \vee T(p,[mn]) \qquad (7.19)$$

$$\neg T(p,m) \vee \neg T(p,[mf(p)1]) \vee T(p,[mn]) \qquad (7.20)$$

Suppose we want to prove the theorem $p \wedge \bigcirc p \wedge \Box(p \rightarrow \bigcirc\bigcirc p) \rightarrow \Box p$. After negation and translation:

(1)	$T(a,m_0)$	
(2)	$T(a,[m_0 1])$	
(3)	$\neg T(a,[m_0 x]) \vee T(a,[m_0 x2])$	
(4)	$\neg T(a,[m_0 b])$	

Here is a refutation:

(5)	$T(a \wedge \bigcirc a, m_0)$	(1), (2), (7.17), (7.1)[16]
(6)	$T(a \wedge \bigcirc a,[m_0 f(a \wedge \bigcirc a)]) \vee T(a \wedge \bigcirc a,[m_0 n])$	(5), (7.19)
(7)	$T(a,[m_0 f(a \wedge \bigcirc a)])$	(6) S-normalized, (4)[17]
(8)	$T(a,[m_0 f(a \wedge \bigcirc a)1])$	
(9)	$\neg T(a \wedge \bigcirc a,[m_0 f(a \wedge \bigcirc a)1]) \vee T(a \wedge \bigcirc a,[m_0 n])$	(5), (7.20)
(10)	$\neg T(a,[m_0 f(a \wedge \bigcirc a)1]) \vee \neg T(a,[m_0 f(a \wedge \bigcirc a)2])$	S-normalized, (4)
(11)	$\neg T(a,[m_0 f(a \wedge \bigcirc a)2])$	(10), (8)
(12)	$\neg T(a,[m_0 f(a \wedge \bigcirc a)])$	(11), (3).2
(13)	empty clause	(12), (7).

Together with T-encoded semantics definitions, T-encoded Hilbert systems yield first-order clause sets which can in principle be given to a FO theorem prover. A lot more efficiency, however, can be gained if extra mechanisms are integrated into the theorem prover [Ohlbach 1998*a*]. The interplay between inference steps and simplification steps using the equivalences as rewrite rules must be carefully controlled. T-encoded Hilbert axioms and rules can be quite naturally turned into theory resolution rules and even recursive applications of the theorem prover itself.

[16]This step actually consists of several steps. First of all, $T(a,[m_0 1])$ is turned into $T(\bigcirc a, m_0)$ using (7.17). Then $T(a,m_0)$ and $T(\bigcirc a,m_0)$ are comprised into $T(a \wedge \bigcirc a, m_0)$ using the T-encoded semantics of $\wedge$ (7.1). A heuristics for triggering these steps is that in the clauses (7.19) and (7.20) a predicate variable occurs in a Skolem function. This gives rise to the fact that a conjunction of instances of the second literal, for example $T(a,[mf(a)]) \wedge T(b,[mf(b)])$, is different to $T(a \wedge b,[mf(a \wedge b)])$. Therefore one should guide the application of clauses with predicate variables in Skolem functions such that conjunctions appear at the term level.

[17]The second literal, $T(a \wedge \bigcirc a,[m_0 n])$ is S-normalized to $T(a,[m_0 n])$ and $T(\bigcirc a,[m_0 n])$. $T(a,[m_0 n])$ is resolved against $\neg T(a,[m_0 b])$, which leaves the first literal of (6) to be S-normalized.

7.2. Indirect translations

We have already seen two examples of indirect translations of one modal logic into another one: the translation from classical modal logic to bi-modal logic given towards the end of Example 4.11, and the layered translation discussed on page 1438. The first results concerning simulations of modal logics in terms of other modal logics were results of Thomason [1974, 1975], where encodings are used to obtain substantial negative results. Thomason shows how one can encode multi-modal logics (and even second-order logic) in uni-modal logic, and obtains incompleteness results and failure of the finite model property. Kracht and Wolter [1999] define encodings of non-normal modal logics in terms of polymodal ones to obtain a series of positive results on axiomatic completeness for non-normal modal logics.

Below, we consider translations from one uni-modal logic into another uni-modal logic, with the aim of putting tools that are available for the latter to work for the former. In particular, we consider the translation from **S4** into **T** as proposed by Cerrito and Cialdea Mayer [1997]. In **S4**, the 4-axiom $\Box p \to \Box\Box p$ allows us to expand $\Box$-formulae to obtain arbitrarily long sequences of $\Box$-operators. With the help of the **T**-axiom $\Box p \to p$, on the other hand, we may delete $\Box$-operators at will.

The idea of the **S4** to **T** translation is to expand $\Box$-formulae sufficiently enough, such that during the proof search one can get the number of nested $\Box$-operators that are really needed by using the **T**-axiom to delete the superfluous $\Box$-operators. This technique must be modified somewhat for **K4**, for which the **T**-axiom is not available. In this case, all formulae $\Box A$ must be replaced by a conjunction $\Box A \wedge \Box\Box A \wedge \cdots \wedge \Box^n A$, which yields very large formulae.

The problem is to determine the maximum number of nested $\Box$-operators which will be needed in the proof, just by inspecting the formula. Cerrito and Cialdea Mayer [1997] found this number by analyzing how often the 4-axiom can be used in a tableaux refutation of a formula A in *negation normal form.* They found that $n \cdot (p+1)$ nested $\Box$-operators are sufficient, where p is the number of $\Box$-subformulae and n is the number of $\Diamond$-subformulae of A.

The precise definition of their translation $Tr_{\mathbf{S4},\mathbf{T}}(A, n)$ with a formula A and a natural number n as arguments is:

$$\begin{aligned}
Tr_{\mathbf{S4},\mathbf{T}}(p,n) &= p \quad \text{if } p \text{ is a literal}\\
Tr_{\mathbf{S4},\mathbf{T}}(A \wedge B,n) &= Tr_{\mathbf{S4},\mathbf{T}}(A,n) \wedge Tr_{\mathbf{S4},\mathbf{T}}(B,n)\\
Tr_{\mathbf{S4},\mathbf{T}}(A \vee B,n) &= Tr_{\mathbf{S4},\mathbf{T}}(A,n) \vee Tr_{\mathbf{S4},\mathbf{T}}(B,n)\\
Tr_{\mathbf{S4},\mathbf{T}}(\Box A,n) &= \Box^n\, Tr_{\mathbf{S4},\mathbf{T}}(A,n)\\
Tr_{\mathbf{S4},\mathbf{T}}(\Diamond A,n) &= \Diamond\, Tr_{\mathbf{S4},\mathbf{T}}(A,n).
\end{aligned}$$

The soundness and completeness theorem for this translation says that A is **S4**-satisfiable if and only if $Tr_{\mathbf{S4},\mathbf{T}}(A, n \cdot (p+1))$ is **T**-satisfiable. Cerrito and Cialdea Mayer's technique compiles the 4-axiom into the formula so that it will not be needed during proof search.

Demri and Goré [1998] have applied this idea to the provability logic **Grz** (for

$$\begin{array}{rcl}
\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(p,i) & = & p \quad \text{if } p \text{ is a literal} \\
\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(\neg A,i) & = & \neg\,\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,1-i) \\
\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A\wedge B,i) & = & \mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(B,i)\wedge \mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(B,i) \\
\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A\rightarrow B,1) & = & \mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,0)\rightarrow \mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(B,1) \\
\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A\rightarrow B,0) & = & \mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,1)\rightarrow \mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(B,0) \\
\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(\Box A,1) & = & \Box(\Box(\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,1)\rightarrow \Box\,\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,0))\rightarrow \\
 & & \qquad \mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,1)) \\
\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(\Box A,0) & = & \Box\,\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,0).
\end{array}$$

Table 9: $\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}$

Grzegorczyk). **Grz** is an extension of **S4** with the Grz-axiom

(Grz) $\vdash \Box(\Box(A\rightarrow\Box A)\rightarrow A)\rightarrow\Box A.$

This axiom cannot be characterized by a first-order property (see [Blackburn et al. 2001, Chapter 3]). Therefore, the usual translations into FO are not applicable. Demri and Goré's translation function $\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,i)$ works for formulae A which need not be in negation normal form; see Table 9. The second argument i is merely used to record the polarity of the subformula.

Exploiting the properties of a cut-free Gentzen calculus for **Grz**, Demri and Goré could prove that a formula A is a **Grz**-theorem if and only if $\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}(A,1)$ is a **S4**-theorem.

The recursive nature of the translation rule for $\Box A$ can cause an exponential blow-up. This can be avoided using standard renaming techniques. Complex subformulae which might get duplicated repeatedly are replaced by new predicate symbols. Adding the definition of these predicate symbols to the original formula usually does not cause a more than polynomial blow-up. By combining the translations $\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}$, $\mathit{Tr}_{\mathbf{S4},\mathbf{T}}$ and ST_m, with suitable renamings of subformulae, Demri and Goré finally obtained a translation from **Grz** into FO such that the size of the translated formulae is $\mathcal{O}((n\log n)^3)$ times the size of the original formula.

Indirect translations like $\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}$ and $\mathit{Tr}_{\mathbf{S4},\mathbf{T}}$ depend very much on the proof theoretic properties of the logic. There is no simple recipe for developing indirect translations. The general idea is to compile Hilbert axioms into the translated formulae. Slight changes to the logic itself, however, may make the translation impossible. Nevertheless, as $\mathit{Tr}_{\mathbf{Grz},\mathbf{S4}}$ has shown, this technique may help considerably in applying FO proof techniques to complex logics with higher-order features (cf. also [Herzig 1990] which contains a translation from the provability logic **G** into **K4**).

7.3. Translations into SnS

In the previous section we managed to reason with an essentially second-order logic such as **Grz** by first-order means. The techniques were very specific, however. The aim of this section is to explore a more general technique for translation-based reasoning with modal logics that escape the first-order realm: translating into the monadic second-order theory of n successor functions. The big advantage of the approach is the large expressive power of the target logic: it allows us to obtain decidability results and reasoning procedures for a large class of modal logics.

The set of monadic second-order logic (MSO) formulae includes all atomic formulae $s = t$, and $X(s)$, where s and t are terms and X is a unary set variable. MSO formulae are closed under the usual boolean connectives, first-order quantifiers over individual variables ($\exists x$, $\forall x$), and second-order quantifiers over the set variables ($\exists X$, $\forall X$). Let $\mathcal{T}_n$ denote the structure of n ($1 \leq n \leq \omega$) successor functions; it is the infinite, uniformly n-branching tree, where the i-th successor function leads to the i-th daughter of a node.

By SnS we denote the monadic second-order theory of n successor functions, and by WSnS we denote the *weak* monadic second-order theory of n successor functions, where the set variables are constrained to range over finite sets only. The decidability of WSnS is due to Thatcher and Wright [1968] and Doner [1970]. The decidability of SnS is known as *Rabin's Tree Theorem* [Rabin 1969]. The decidability of WSnS is based on a close correspondence between formulae in WSnS and finite automata. More precisely, any relation definable in WS2S can also be defined by a tree automaton that encodes the satisfying assignments to the formula in the labels on the nodes of the tree that it accepts. The coding is simple: assignment functions map first-order variables to nodes in $\mathcal{T}_2$ and second-order variables to sets of nodes in $\mathcal{T}_2$. The labeling functions on the trees accepted by the automaton do the opposite: they map nodes to the set of variables to which that is assigned.

When dealing with *weak* S2S, all sets are finite, so we can restrict ourselves to automata over finite trees; since efficient minimization techniques exists for finite trees, this opens the way to the development of automated reasoning tools, at least in principle. But there are some challenging difficulties. The correspondence between WS2S and automata is an inductive one, that follows the construction of formulae. Now, the negation construction on automata only works for deterministic automata, while the projection function (which implements quantification) yields *non*-deterministic automata. Using the subset construction, tree automata may be determinized, but at the cost of an exponential blow-up. Nevertheless, the logic-automata connection may be used successfully for a number of purposes.

For a start, it provides a powerful tool for establishing decidability results in nonclassical logic. Rabin's Tree Theorem was applied in modal logic almost immediately: Fine [1970] 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 [1971*a*, 1971*b*, 1971*c*] applied it to a wide range of modal logics in many different languages. These papers are essential reading for readers interested in the method, though before tackling them, it's probably best to first see what Gabbay, Hodkinson and Reynolds [1994] have to say on the subject.

Secondly, as indicated above, the decision procedure for WS2S is semantically based: it translates a WS2S formula A to an automaton $\mathcal{A}_A$ that recognizes valuations verifying A. The MONA system [Henriksen, Jensen, Jørgensen, Klarlund, Paige, Rauhe and Sandhol 1995] implements this decision procedure. Input to MONA is a script consisting of a sequence of definitions followed by a formula A to be proved. MONA computes $\mathcal{A}_A$ and, depending on the result, declares A to be valid or delivers a counter example. Despite the non-elementary worst-case complexity of WS2S, MONA works well in practice on a large range of problems; Basin and Klarlund [1998] 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 nonclassical logics such as propositional dynamic logic.

7.4. Translations into weak set theories

The final approach towards translation-based automated reasoning for modal logic that we want to mention here is due to Montanari, Policriti, and their colleagues and students. In [D'Agostino, Montanari and Policriti 1995] they propose a translation of modal logic into (weak) set theories that works for all normal complete finitely axiomatizable modal logics, not just for the ones that are complete with respect to a first-order definable class of structures. In particular, the method is also applicable if the modal logic at hand is specified with Hilbert axioms only.

The basic idea is to represent a Kripke frame as a set, with the accessibility relation modeled using the membership relation $\in$. For computational reasons the set theory that axiomatizes $\in$ should be a finitely (first-order) axiomatizable theory Ω. Given a modal formula $A(p_1, \ldots, p_n)$, its translation as a set-theoretic term with variables x, x_1, ..., x_n is written as $A^*(x, x_1, \ldots, x_n)$. The latter represents the set of states in the frame x in which the formula A holds.

To make things precise, we start by specifying the theory Ω; its (first-order) language contains relation symbols $\in$, $\subseteq$, and function symbols $\cup$, $\setminus$, and Pow (the power set operation). The axioms are $x \in y \cup z \leftrightarrow x \in y \vee x \in z$; $x \in y \setminus z \leftrightarrow x \in y \wedge x \notin z$; $x \subseteq y \leftrightarrow \forall z\,(z \in x \rightarrow z \in y)$; and $x \in Pow(y) \leftrightarrow x \subseteq y$. Observe that this theory Ω lacks both the extensionality and foundation axioms found in traditional set theories; in fact, the authors prefer to work with set-theoretic universes satisfying the *anti*-foundation axiom AFA [Aczel 1988].

Now, the translation $(\cdot)^*$ is defined as follows:

$$\begin{array}{rclrcl} p_i^* & := & x_i & (A \vee B)^* & := & A^* \cup B^* \\ (\neg A)^* & := & x \setminus A^* & (\Box A)^* & := & Pow(A^*), \end{array} \qquad (\dagger)$$

where x is a variable different from x_i $(i = 1, \ldots, n)$.

7.7. Theorem. *For any finitely axiomatizable modal logic* $\mathbf{L} = \mathbf{K} + B(\alpha_1, \ldots, \alpha_m)$, *where* $B(\alpha_1, \ldots, \alpha_m)$ *is an axiom schema, the following are equivalent:*

1. $\vdash_{\mathbf{L}} A(p_1, \ldots, p_n)$
2. $\Omega \vdash \forall x\, (\mathit{Trans}(x) \wedge \mathit{Axiom}_{\mathbf{L}}(x) \rightarrow \forall x_1 \ldots \forall x_n\, (x \subseteq A^*(x, x_1, \ldots, x_n)))$.

Here, $\mathit{Trans}(x)$ is short for $\forall y\, (y \in x \rightarrow y \subseteq x)$, and $\mathit{Axiom}_{\mathbf{L}}(x)$ is short for $\forall y_1 \ldots \forall y_m\, (x \subseteq B^*(x, y_1, \ldots, y_m))$.

Instead of translating Hilbert axioms, one may use a semantics for **L** whenever it is available. Furthermore, the method is easily extended to poly-modal logics. As to actual reasoning with the set theories into which modal logics are translated, D'Agostino et al. [1995] consider the use of theory resolution; to this end, skolemized versions of Ω must be decidable [Policriti and Schwartz 1992]. At the time of writing, experimental results comparing the approach to other translation-based approaches to automated reasoning in modal logic are not available.

Van Benthem, D'Agostino, Montanari and Policriti [1997] extend the connection between modal logic and set theory outlined above to capture a larger part of the non r.e. notion of modal logical consequence. The notion of validity on so-called *general frames* [Blackburn et al. 2001, Chapter 5] — and hence modal derivability in its full strength — can be captured by modifying the translation (†) without changing the underlying set theory Ω. To deal with so-called *extended modal logics* [de Rijke 1993], there is a further proposal to enrich the underlying set theory Ω containing, essentially, the Gödel constructible functions that allows one to capture weak monadic second-order logic [van Benthem, D'Agostino, Montanari and Policriti 1998].

8. Conclusion

In this chapter we have discussed various ways of translating nonclassical logics into first-order predicate logic. We have tried to explain the basic ideas and principles to help the reader understand and apply the ideas to his or her own logic.

Nonclassical logics presented as Hilbert systems can be translated into FO by encoding the formulae as FO-terms. This is the easiest and most flexible way of encoding a nonclassical logic in FO. Unfortunately, FO theorem provers are extremely inefficient for this kind of encoding.

Many familiar nonclassical logics, however, enjoy a sound and complete possible worlds semantics. We have shown how to derive a relational translation directly from their semantics. The main purpose of the translation idea is to use the results, tools, and techniques from the target logic, in particular theorem provers, for proving translated theorems. Unfortunately, standard inference techniques for FO do not automatically provide decision procedures for the corresponding fragments. To overcome this we presented modifications of the translation method.

The layered relational translation encodes a very strong form of the tree model property; it can be used as a preprocessing technique for existing tools. Resolution (with factoring and condensing) refined by a very simple ordering may be used as a decision procedure for the resulting modal fragment.

Another, more dramatic, modification of the relational translation starts from the observation that binary relations can be decomposed into sets of functions. In the resulting functional translation, reasoning about accessible worlds is incorporated into the unification algorithm. Resolution becomes a decision procedure without any special strategies. Specifying frame classes in the functional style may introduce equations. This is an advantage if the equations can be turned into a theory unification algorithm, and the algorithm can actually be incorporated in a prover. Unfortunately, only the developer of a theorem prover is usually able to do this.

To overcome this problem we introduced the semi-functional translation. In the case of modal logic it translates the $\Diamond$-operator functionally and the $\Box$-operator relationally. The characteristic frame properties are specified relationally. Using saturation techniques one can simplify the characteristic frame axioms considerably. In the end we achieved the best of both kinds of translations: small translated formulae and the possibility to do a considerable part of the reasoning about accessible worlds within the (standard) unification algorithm, and an optimized treatment of the characteristic frame axioms without having to change the implementation of the unification algorithm.

Finally, we presented a few examples of indirect translations for modal logics where the Hilbert axioms are compiled into modal formulae. This way one can even translate logics with second-order properties into first-order logic. The techniques, however, are very specific to the logics, and require a detailed analysis of a Gentzen or sequent type proof system. More widely applicable solutions for logics with second-order properties are provided by translations into SnS, or into weak set theories.

In the course of this chapter we have touched on many open issues, ranging from examining the performance of provers on (translated) nonclassical logics other than familiar modal logics (page 1440), to the connection between Segerberg's cluster-based analysis of modal logics and saturating background theories in the semi-functional translation approach (page 1464).

The big issue that drives the field as a whole, however, is how we can encode, by syntactic means, the restricted nature of source logics inside the target logic. We have seen several proposals, formulated in terms of special fragments such as finite variable fragments or guarded fragments, or in terms of modified translations such as the layered, functional, and semi-functional translation. It is not the actual syntax that matters here. The satisfiability problem for the logics we consider is usually PSPACE-hard or worse [Ladner 1977, Fagin et al. 1995, Blackburn et al. 2001]. Therefore, the performance of satisfiability checkers depends more on the strategies and heuristics than on the actual syntax, whether it is the original or the translated one. Different syntactic presentations, however, may give more or less easy access to the information which is relevant for strategies and heuristics to guide the search. What really matters is the right combination of the syntactic presentation of the satisfiability problem and the corresponding search control mechanisms. For the time being there is no clear winner among the different approaches.

To conclude the chapter, we return to an issue raised in the introduction to this chapter: at the end of the day, what should one use: direct, purpose-built proof tools for nonclassical logics, or indirect, translation-based ones? The problem of finding efficient proof procedures for the translated formulae has not yet been solved satisfactorily. The main reason for this is that the original motivation for developing translation methods came from the desire to use existing predicate logic theorem provers for proving theorems in nonclassical logics *without* making any changes to the implementation of the theorem prover. This works, but only to a certain extent. Experience has shown that we need special, purpose-built refinements to complement our translations if we are going to keep up with dedicated tools, and more importantly, if we are going to get acceptable performance.

Acknowledgments

We would like to thank the editors for their patience. Maarten de Rijke was supported by the Spinoza project 'Logic in Action' and by a grant from the Netherlands Organization for Scientific Research (NWO), under project number 365-20-005.

Bibliography

Aczel P. [1988], *Non-Well-Founded Sets*, CSLI Publications.

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

Andrews P. [2001], Classical type theory, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 15, pp. 965–1007.

Areces C. and de Rijke M. [2000], Description and/or hybrid logic, *in* 'Proceedings AiML-ICTL 2000'.

Areces C., Gennari R., Heguiabehere J. and de Rijke M. [2000], Tree-based heuristics for modal theorem proving, *in* W. Horn, ed., 'Proceedings ECAI-2000', IOS Press.

Auer P. [1992], Unification with associative functions, PhD thesis, Technical University Vienna.

Auffray Y. and Enjalbert P. [1992], 'Modal theorem proving: An equational viewpoint', *Journal of Logic and Computation* **2**(3), 247–297.

Baader F. and Snyder W. [2001], Unification theory, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 8, pp. 445–532.

Baaz M., Egly U. and Leitsch A. [2001], Normal form transformations, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 5, pp. 273–333.

Baaz M., Fermüller C. and Leitsch A. [1994], A non-elementary speed up in proof length by structural clause transformation, *in* 'Proceedings LICS'94: Logic in Computer Science', IEEE Computer Society, pp. 213–219.

Baaz M., Fermüller C. and Salzer G. [2001], Automated deduction for many-valued logics, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 20, pp. 1355–1402.

Bachmair L. and Ganzinger H. [2001], Resolution theorem proving, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 2, pp. 19–99.

Barendregt H. and Geuvers H. [2001], Proof-assistants using dependent type systems, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 18, pp. 1149–1238.

Basin D. and Klarlund N. [1998], 'Automata based symbolic reasoning in hardware verification', *Journal of Formal Methods in Systems Design* **13**, 255–288.

Blackburn P., de Rijke M. and Venema Y. [2001], *Modal Logic*, Cambridge University Press.

Bli [2000]. Bliksem Version 1.10B. URL: `http://www.mpi-sb.mpg.de/~bliksem/`. Accessed Jan. 16, 2000.

Bockmayr A. and Weispfenning V. [2001], Solving numerical constraints, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 12, pp. 749–842.

Bundy A. [2001], The automation of proof by mathematical induction, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 13, pp. 845–911.

Calvanese D., Giacomo G. D., Lenzerini M. and Nardi D. [2001], Reasoning in expressive description logics, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 23, pp. 1581–1634.

Cerrito S. and Cialdea Mayer M. [1997], 'A polynomial translation of **S4** into **T** and contraction-free tableaux for **S4**', *Logic Journal of the IGPL* **5**(2), 287–300.

Chellas B. F. [1980], *Modal Logic: An Introduction*, Cambridge University Press, Cambridge.

Chou S. C. and Gao X. S. [2001], Automated reasoning in geometry, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 11, pp. 705–747.

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

Clarke E. and Schlingloff H. [2001], Model checking, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 24, pp. 1635–1790.

Comon H. [2001], Inductionless induction, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 14, pp. 913–962.

D'Agostino G., Montanari A. and Policriti A. [1995], 'A set-theoretic translation method for polymodal logics', *Journal of Automated Reasoning* **15**, 317–337.

Davis M. [2001], The early history of automated deduction, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 1, pp. 3–15.

de Nivelle H. [1998], A resolution decision procedure for the guarded fragment, *in* C. Kirchner and H. Kirchner, eds, 'Proceedings CADE-15: International Conference on Automated Deduction', Vol. 1421 of *LNAI*, Springer-Verlag, pp. 191–204.

de Nivelle H. [1999], A new translation method for **S4**. Manuscript.

de Nivelle H. and de Rijke M. [to appear], 'Deciding the guarded fragments by resolution', *Journal of Symbolic Computation* .

de Rijke M. [1993], Extending Modal Logic, PhD thesis, ILLC, University of Amsterdam.

de Rijke M. [1999], Restricted description languages. Manuscript, ILLC, University of Amsterdam.

Degtyarev A. and Voronkov A. [2001*a*], Equality reasoning in sequent-based calculi, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 10, pp. 609–704.

Degtyarev A. and Voronkov A. [2001*b*], The inverse method, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 4, pp. 179–272.

Demri S. and Goré R. [1998], An $\mathcal{O}((n \log n)^3)$-time transformation from *Grz* into decidable fragments of classical first-order logic, *in* 'Proceedings FTP'98: International Workshop on First-Order Theorem Proving', TU-Wien Technical Report E1852-GS-981, pp. 127–134.

Dershowitz N. and Plaisted D. [2001], Rewriting, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 9, pp. 533–608.

Dix J., Furbach U. and Niemelä I. [2001], Nonmonotonic reasoning: Towards efficient calculi and implementations, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 19, pp. 1241–1354.

DONER J. [1970], 'Tree acceptors and some of their applications', *Journal of Computer and System Sciences* **4**, 406–451.

DOWEK G. [2001], Higher-order unification and matching, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 16, pp. 1009–1062.

EMERSON E. AND HALPERN J. [1985], 'Decision procedures and expressiveness in the temporal logic of brnaching time', *Journal of Computer and System Sciences* **30**, 1–24.

FAGIN R., HALPERN J., MOSES Y. AND VARDI M. [1995], *Reasoning About Knowledge*, The MIT Press.

FARIÑAS DEL CERRO L. AND HERZIG A. [1988], Linear modal deductions, *in* E. Lusk and R. Overbeek, eds, 'Proceedings CADE'88: International Conference on Automated Deduction', Springer.

FERMÜLLER C., LEITSCH A., HUSTADT U. AND TAMMET T. [2001], Resolution decision procedures, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 25, pp. 1791–1849.

FERMÜLLER C., LEITSCH A., TAMMET T. AND ZAMOV N. [1993], *Resolution Methods for the Decision Problem*, Vol. 679 of *LNAI*, Springer.

FINE K. [1970], 'Propositional quantifiers in modal logic', *Theoria* **36**, 331–346.

GABBAY D. [1971*a*], 'Decidability results in non-classical logics', *Annals of Mathematical Logic* **10**, 237–285.

GABBAY D. [1971*b*], 'On decidable, finitely axiomatizable modal and tense logics without the finite model property I', *Israel Journal of Mathematics* **10**, 478–495.

GABBAY D. [1971*c*], 'On decidable, finitely axiomatizable modal and tense logics without the finite model property II', *Israel Journal of Mathematics* **10**, 496–503.

GABBAY D. [1981*a*], Expressive functional completeness in tense logic (preliminary report), *in* U. Mönnich, ed., 'Aspects of Philosophical Logic', Reidel, pp. 91–117.

GABBAY, D. AND GUENTHNER, F., EDS [1984], *Handbook of Philosophical Logic*, Vol. 2, Reidel, Dordrecht.

GABBAY D., HODKINSON I. AND REYNOLDS M. [1994], *Temporal Logic: Mathematical Foundations and Computational Aspects*, Oxford University Press.

GABBAY D. M. [1981*b*], An irreflexivity lemma with applications to axiomatizations of conditions on tense frames, *in* U. Monnich, ed., 'Aspects of Philosophical Logic', Reidel, pp. 67–89.

GABBAY D. M. [1996], *Labelled Deductive Systems; Principles and Applications. Vol 1: Introduction*, Vol. 126, Oxford University Press.

GABBAY D. M. [1999], *Fibring Logics*, Oxford University Press.

GANZINGER H. AND DE NIVELLE H. [1999], A superposition decision procedure for the guarded fragment with equality, *in* 'Proceedings LICS'99: Logic in Computer Science', IEEE Computer Society, pp. 295–304.

GANZINGER H., MEYER C. AND VEANES M. [1999], The two-variable guarded fragment with transitive relations, *in* 'Proceedings LICS'99: Logic in Computer Science', IEEE Computer Society, pp. 24–34.

GASQUET O. AND HERZIG A. [1993], Translating non-normal modal logics into normal modal logics, *in* I. J. Jones and M. Sergot, eds, 'Proceedings DEON-94: International Workshop on Deontic Logic in Computer Science', TANO, Oslo.

GIUNCHIGLIA E., GIUNCHIGLIA F. AND TACCHELLA A. [2000], 'SAT-based decision procedures for classical modal logics', pp. 403–426.

GÖDEL K. [1932], 'Ein Spezialfall des Entscheidungsproblem der theoretischen Logik', *Ergebn. math. Kolloq.* **2**, 113–115.

GOLDBLATT R. [1992], *Logics of Time and Computation*, CSLI Publications.

GRÄDEL E. [2000], 'On the restraining power of guards', *Journal of Symbolic Logic* . to appear.

GRÄDEL E., OTTO M. AND ROSEN E. [1997*a*], Two-variable logic with counting is decidable, *in* 'Proceedings LICS'97: Logic in Computer Science', IEEE Computer Society, pp. 306–317.

GRÄDEL E., OTTO M. AND ROSEN E. [1997*b*], Undecidability results on two-variable logics, *in* 'Proceedings STACS'97', Vol. 1200 of *LNCS*, Springer, pp. 249–260.

GRÄDEL E. AND WALUKIEWICZ I. [1999], Guarded fixed point logic, *in* 'Proceedings LICS'99: Logic in Computer Science', IEEE Computer Society, pp. 45–54.

GRAF P. [1996], *Term Indexing*, Vol. 1053 of *LNAI*, Springer.

HÄHNLE R. [2001], Tableaux and related methods, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 3, pp. 100–178.

HÁJEK P. AND PUDLÁK P. [1991], *Metamathematics of First-Order Arithmetic*, Perspectives in Mathematical Logic, Springer.

HALPERN J. [1995], 'The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic', *Artificial Intelligence* **75**, 361–372.

HAREL D. [1984], Dynamic logic, *in* Gabbay and Guenthner [1984], pp. 497–604.

HENKIN L. [1967], 'Logical systems containing only a finite number of symbols'. Séminiare de Mathématique Supérieures 21, Les Presses de l'Université de Montréal, Montréal.

HENRIKSEN J., JENSEN J., JØRGENSEN M., KLARLUND N., PAIGE R., RAUHE T. AND SANDHOL A. [1995], MONA: Monadic second-order logic in practice, *in* 'Proceedings TACAS'95', LNCS, Springer, pp. 479–506.

HERZIG A. [1989], Raisonnement Automatique en Logique Modale et Algorithmes d'unification., PhD thesis, Université Paul-Sabatier, Toulouse.

HERZIG A. [1990], 'A translation from propositional modal logic G into K4', Université Paul Sabatier, Toulouse.

HORROCKS I., PATEL-SCHNEIDER P. AND SEBASTIANI R. [2000], 'An analysis of emperical testing for modal decision procedures', *Logic Journal of the IGPL* **8**(3), 293–323.

HUSTADT U. [1999], Resolution-Based Decision Procedures for Subclasses of First-Order Logic, PhD thesis, Max-Planck Institute for Computer Science, Saarbrücken.

HUSTADT U., SCHMIDT R. AND WEIDENBACH C. [1998], Optimised functional translation and resolution, *in* 'Proceedings Tableaux'98', LNCS, Springer, pp. 36–37.

IMMERMAN N. AND KOZEN D. [1987], Definability with bounded number of bound variables, *in* 'Proceedings LICS'87: Logic in Computer Science', IEEE Computer Society, Washington, pp. 236–244.

JANIN D. AND WALUKIEWICZ I. [1996], On the expressive completeness of the propositional μ-calculus w.r.t. monadic second-order logic, *in* 'Proceedings CONCUR'96', pp. 263–277.

JOYNER JR. J. [1976], 'Resolution strategies as decision procedures', *Journal of the ACM* **23**, 398–417.

KALMÁR L. [1933], 'Über die Erfüllbarkeit derjenigen Záhlausdrücke, welche in der Normalform zwei benachbarte Allzeichen enthalten', *Math. Annal.* **108**, 466–484.

KRACHT M. AND WOLTER F. [1999], 'Normal monomodal logics can simulate all others', *Journal of Symbolic Logic* **64**, 99–138.

KURTONINA N. AND DE RIJKE M. [1997], 'Bisimulations for temporal logic', *Journal of Logic, Language and Information* **6**, 403–425.

KURTONINA N. AND DE RIJKE M. [1999], 'Expressiveness of concept expressions in first-order description logics', *Artificial Intelligence* **107**, 303–333.

LADNER R. [1977], 'The computational complexity of provability in systems of modal logic', *SIAM Journal on Computing* **6**, 467–480.

LETZ R. AND STENZ G. [2001], Model elimination and connection tableau procedures, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 28, pp. 2015–2114.

ŁUKASIEWICZ J. [1970], *Selected Works*, North Holland. Edited by L. Borkowski.

MCCUNE W. AND WOS L. [1992], Experiments in automated deduction with condensed detachment, *in* D. Kapur, ed., 'Proceedings CADE-11', Vol. 607 of *LNAI*, Springer Verlag, pp. 209–223.

Mints G. [1989], 'Resolution calculi for modal logics', *Amer. Math. Soc. Transl.* **143**, 1–14.

Moller F. and Rabinovich A. [1999], On the expressive power of CTL*, *in* 'Proceedings LICS'99: Logic in Computer Science', IEEE Computer Society, pp. 360–369.

Mortimer M. [1975], 'On languages with two variables', *Zeitschr. f. math. Logik u. Grundlagen d. Math.* **21**, 135–140.

Nieuwenhuis R. and Rubio A. [2001], Paramodulation-based theorem proving, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 7, pp. 371–443.

Nonnengart A. [1992], First-order modal logic theorem proving and standard PROLOG, Technical Report MPI-I-92-228, Max-Planck-Institute for Computer Science, Saarbrücken, Germany.

Nonnengart A. [1993], First-order modal logic theorem proving and functional simulation, *in* R. Bajcsy, ed., 'Proceedings IJCAI-13', Vol. 1, Morgan Kaufmann Publishers, pp. 80–85.

Nonnengart A. [1995], A Resolution-Based Calculus for Temporal Logics, PhD thesis, Universität des Saarlandes, Saarbrücken, Germany.

Nonnengart A. [1996], Resolution-based calculi for modal and temporal logics, *in* S. McRobbie, ed., 'Proceedings CADE-13: International Conference on Automated Deduction', Vol. 1104 of *LNAI*, Springer Verlag, pp. 598–612.

Nonnengart A. and Weidenbach C. [2001], Computing small clause normal forms, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 6, pp. 335–367.

Ohlbach H. J. [1988*a*], A resolution calculus for modal logics, *in* E. Lusk and R. Overbeek, eds, 'Proceedings CADE-88: International Conference on Automated Deduction', Vol. 310 of *LNCS*, Springer-Verlag, pp. 500–516.

Ohlbach H. J. [1988*b*], A resolution calculus for modal logics, SEKI Report SR-88-08, FB Informatik, Universität Kaiserslautern, Germany. PhD Thesis, short version appeared in [Ohlbach 1988*a*].

Ohlbach H. J. [1991], 'Semantics based translation methods for modal logics', *Journal of Logic and Computation* **1**(5), 691–746.

Ohlbach H. J. [1998*a*], Combining Hilbert-style and semantic reasoning in a resolution framework, *in* C. Kirchner and H. Kirchner, eds, 'Proceedings CADE-15: International Conference on Automated Deduction', Vol. 1421 of *LNAI*, Springer Verlag, pp. 205–219.

Ohlbach H. J. [1998*b*], 'Elimination of self-resolving clauses', *Journal of Automated Reasoning* **20**(3), 317–336.

Ohlbach H. J. and Schmidt R. A. [1997], 'Functional translation and second-order frame properties of modal logics', *Journal of Logic and Computation* **7**(5), 581–603.

Ohlbach H. J. and Wrightson G. [1984], Solving a problem in relevance logic with an automated theorem prover, *in* R. E. Shostak, ed., 'Proceedings CADE-7: International Conference on Automated Deduction', Vol. 170 of *LNCS*, Springer Verlag, pp. 496–508.

Ohlbach H., Nonnengart A., de Rijke M. and Gabbay D. [2001], Encoding two-valued nonclassical logics in classical logic, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 21, pp. 1403–1486.

Otto M. [1997], *Bounded Variable Logics and Counting — A Study in Finite Models*, Vol. 9 of *Lecture Notes in Logic*, Springer.

Patel-Schneider P. [1998], DLP system description, *in* 'Proceedings DL'98', pp. 87–89. URL: `http://suncite.informatik.rwth-aachen.de/Publications/CEUR-WS`.

Paulson L. C. [1994], *Isabelle: a Generic Theorem Prover*, Vol. 828 of *LNCS*, Springer Verlag, Berlin.

Pfenning F. [2001], Logical frameworks, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 17, pp. 1063–1147.

Policriti A. and Schwartz J. [1992], T-theorem proving I, Research report 08/92, Università di Udine.

QUINE W. [1953], *From a Logical Point of View*, Harvard Umiversity Press.

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

RAMAKRISHNAN I., SEKAR R. AND VORONKOV A. [2001], Term indexing, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 26, pp. 1853–1964.

ROSEN E. [1997], 'Modal logic over finite structures', *Journal of Logic, Language and Information* **6**, 427–439.

ROUNDS W. [1997], Feature logics, *in* van Benthem and ter Meulen [1997].

SAHLQVIST H. [1975], Completeness and correspondence in the first and second order semantics for modal logics, *in* S. Kanger, ed., 'Proceedings of the 3rd Scandinavian Logic Symposium, 1973', North Holland, Amsterdam, pp. 110–143.

SCHMIDT R. [1997], Optimised Modal Translation and Resolution, PhD thesis, Universität des Saarlandes.

SCHMIDT R. A. [1998*a*], *E*-unification for subsystems of **S4**, *in* 'Proceedings RTA'98: Term Rewriting and Applications', Vol. 1379 of *LNCS*, Springer, pp. 106–120.

SCHMIDT R. A. [1998*b*], Resolution is a decision procedure for many propositional modal logics, *in* M. Kracht, M. de Rijke, H. Wansing and M. Zakharyaschev, eds, 'Advances in Modal Logic, Vol. 1', CSLI Publications, pp. 189–208.

SCHÜTTE K. [1934], 'Untersuchungen zum Entscheidungsproblem der mathematischen Logik', *Math. Annal.* **109**, 572–603.

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

SEGERBERG K. [1971], An essay in classical modal logic, Technical Report 13, University of Uppsala, Filosofiska Studier. Volume 1-3.

TAN [2000]. TANCS: Tableaux Non Classical Systems Comparison. URL: `http://www.dis.uniroma1.it/~tancs/`. Accessed Jan. 16, 2000.

THATCHER J. AND WRIGHT J. [1968], 'Generalized finite automata theory with an application to a decision problem of second-order logic', *Mathematical Systems Theory* **2**, 57–81.

THOMASON S. [1974], 'Reduction of tense logic to modal logic I', *Journal of Symbolic Logic* **39**, 549–551.

THOMASON S. [1975], 'Reduction of tense logic to modal logic II', *Theoria* **41**, 154–169.

VAN BENTHEM J. [1976], Modal Correspondence Theory, PhD thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam.

VAN BENTHEM J. [1983], *Modal Logic and Classical Logic*, Bibliopolis.

VAN BENTHEM J. [1984], Correspondence theory, *in* D. M. Gabbay and F. Guenthner, eds, 'Handbook of Philosophical Logic, Vol. II, Extensions of Classical Logic, Synthese Library Vol. 165', D. Reidel Publishing Company, Dordrecht, pp. 167–248.

VAN BENTHEM J. [1997], Dynamic bits and pieces, Technical Report LP-1997-01, ILLC, University of Amsterdam.

VAN BENTHEM J., D'AGOSTINO G., MONTANARI A. AND POLICRITI A. [1997], 'Modal deduction in second-order logic and set theory I', *Journal of Logic and Computation* **7**, 251–265.

VAN BENTHEM J., D'AGOSTINO G., MONTANARI A. AND POLICRITI A. [1998], 'Modal deduction in second-order logic and set theory II', *Studia Logica* **60**, 387–420.

VAN BENTHEM, J. AND TER MEULEN, A., EDS [1997], *Handbook of Logic and Language*, Elsevier Science.

VARDI M. [1997], Why is modal logic so robustly decidable?, *in* 'DIMACS Series in Discrete Mathematics and Theoretical Computer Science 31', AMS, pp. 149–184.

WAALER A. [2001], Connections in nonclassical logics, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 22, pp. 1487–1578.

WALTHER C. [1987], *A Many-Sorted Calculus Based on Resolution and Paramodulation*, Pitman.

Weidenbach C. [2001], Combining superposition, sorts and splitting, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 27, pp. 1965–2013.

Zamov N. [1989], 'Modal resolutions', *Soviet Math.* **33**, 22–29.

Index