Técnicas de razonamiento automático para lógicas híbridas

Gorín, Daniel Alejandro

2009

UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Técnicas de razonamiento automático para lógicas híbridas

Tesis presentada para optar al título de Doctor de la Universidad de Buenos Aires en el área de Ciencias de la Computación.

Daniel Alejandro Gorín

Directores de tesis: Carlos Areces
Verónica Becher

Buenos Aires, 2009

Contents

Resumen

Técnicas de razonamiento automático para lógicas híbridas

Las "lógicas híbridas" extienden a las lógicas modales tradicionales con el poder de describir y razonar sobre cuestiones de identidad, lo cual es clave para muchas aplicaciones. Aunque lógicas modales que hoy llamaríamos "híbridas" pueden rastrearse hasta cuatro décadas atrás, su estudio sistemático data de fines de la década del '90. Parte de su interés proviene de que llenan un hueco de expresividad importante de las lógicas modales tradicionales.

Uno de los temas de esta tesis es el problema de la satisfacibilidad para la lógica híbrida más conocida, denominada $\mathcal{H}(@, \downarrow)$, y algunas de sus sublógicas. El de la satisfacibilidad es el problema fundamental en razonamiento automático. En el caso de las lógicas híbridas, éste se ha estudiado fundamentalmente a partir del método de tableaux. En esta tesis intentamos completar el panorama del área investigando el problema de la satisfacibilidad para lógicas híbridas usando resolución clásica de primer orden (vía traducciones) y variaciones de un cálculo basado en resolución que opera directamente sobre fórmulas híbridas.

Presentamos, en primer lugar, traducciones de complejidad lineal de fórmulas de $\mathcal{H}(@, \downarrow)$ a lógica de primer orden, que preservan satisfacibilidad. Éstas están concebidas de manera de reducir el espacio de búsqueda de un demostrador automático basado en resolución de primer orden.

Luego cambiamos nuestra atención a cálculos que operan directamente sobre fórmulas híbridas. En particular, consideramos el cálculo llamado de"resolución directa". Inspirados por el caso clásico, transformamos este cálculo en uno de resolución ordenada con funciones de selección y probamos que posee la "propiedad de reducción de contraejemplos", de lo cual se deduce que es completo y compatible con el criterio de redundancia estándar. Mostramos también que un refinamiento de este cálculo es un método de decisión para la sublógica decidible $\mathcal{H}(@)$.

En la última parte de esta tesis, consideramos ciertas formas normales para lógicas híbridas y otras lógicas modales extendidas. En particular nos interesan formas normales donde se garantice que ciertas modalidades no aparecen por debajo de otros operadores modales. Este tipo de transformaciones puede ser aprovechadas en una etapa de preprocesamiento a fin de reducir el número de inferencias requeridas por un demostrador modal.

Al intentar expresar estos resultados de extractibilidad de una manera que comprenda también otras lógicas modales extendidas, llegamos a una formulación de la semántica modal basada en un tipo novedoso de modelos definidos de manera coinductiva. Muchas lógicas modales extendidas (incluyendo las lógicas híbridas) pueden verse en términos de clases de modelos coinductivos. De esta manera, resultados que antes debían probarse por separado para cada lenguaje (pero cuyas pruebas eran en general rutinarias) pueden establecerse de manera general.

Palabras clave: *lógicas híbridas, demostración automática, traducciones a primer orden, resolución directa, métodos de decisión, modelos coinductivos, formas normales, extractabilidad de modalidades.*

Abstract

Automated reasoning techniques for hybrid logics

Hybrid logics augment classical modal logics with machinery for describing and reasoning about identity, which is crucial in many settings. Although modal logics we would today call "hybrid" can be traced back to the work of Prior in the 1960's, their systematic study only began in the late 1990's. Part of their interest comes from the fact they fill an important expressivity gap in modal logics. In fact, they are sometimes referred to as "modal logics with equality".

One of the unifying themes of this thesis is the satisfiability problem for the arguably best-known hybrid logic, $\mathcal{H}(@, \downarrow)$, and some of its sublogics. Satisfiability is the basic problem in automated reasoning. In the case of hybrid logics it has been studied fundamentally using the tableaux method. In this thesis we attempt to complete the picture by investigating satisfiability for hybrid logics using first-order resolution (via translations) and variations of a resolution calculus that operates directly on hybrid formulas.

We present firstly several satisfiability-preserving, linear-time translations from $\mathcal{H}(@, \downarrow)$ to first-order logic. These are conceived in a way such that they tend to reduce the search space of a resolution-based theorem prover for first-order logic. notations can be safely ignored.

We then move our attention to resolution-based calculi that work directly on hybrid formulas. In particular, we will consider the so-called *direct resolution* calculus. Inspired by first-order logic resolution, we turn this calculus into a calculus of ordered resolution with selection functions and prove that it possesses the *reduction property for counterexamples* from which it follows its completeness and that it is compatible with the well-known standard redundancy criterion. We also show that certain refinement of this calculus constitutes a decision procedure for $\mathcal{H}(@)$, a decidable fragment of $\mathcal{H}(@, \downarrow)$.

In the last part of this thesis we investigate certain normal forms for hybrid logics and other extended modal logics. We are interested in normal forms where certain modalities can be guaranteed not to occur under the scope of other modal operators. We will see that these kind of transformations can be exploited in a pre-processing step in order to reduce the number of inferences required by a modal prover.

In an attempt to formulate these results in a way that encompasses also other extended modal logics, we arrived at a formulation of modal semantics in terms of a novel type of models that are coinductively defined. Many extended modal logics (such as hybrid logics) can be defined in terms of classes of coinductive models. This way, results that had to be proved separately for each different language (but whose proofs were known to be mere routine) now can be proved in a general way.

Keywords: *hybrid logics, automated reasoning, first-order translations, direct resolution, decision methods, coinductive models, normal forms, extractability of modalities.*

Acknowledgments

A lot of people helped me out, one way or the other (even without noticing it!), throughout all the years since I started to work in this thesis. I'm grateful to all of them.

I'd like to express my deepest gratitude to Carlos Areces. I can't imagine having a more supportive, guiding, enlightening and hard-working supervisor. And that is only half of the story. He has been a great tour guide and trip advisor, amazing cook and outstanding host (he used to run the best B&B in the whole Lorraine). I very much enjoyed browsing, discussing and borrowing things from his personal "médiathèque".

I want to thank Verónica Becher for she very strongly encouraged me to start a PhD. It was a difficult decision to make at that time but one I don't regret. I want to thank also Patrick Blackburn for all the interesting discussions; I also enjoyed his lectures, in that very passionate style of his.

I had the pleasure to work with many members of GLyC, the research group I'm part of: Santiago Figueira, Sergio Mera (thesis evil twin), Mariano Pérez Rodríguez, Diego Figueira, Daniel Koile, Facundo Carreiro and Ricardo Rodríguez. I also worked and shared very good moments with many other members of the CS Department at the University of Buenos Aires: Diego Garbervetsky, Esteban Mocskos, Juan Pablo Galeotti, Diego Fernandez Slezak, Enrique Tobis, Fernando Schapachnik, Mariano Moscato, Marina Groshaus, Guido de Caso, Sergio Daicz, Pablo Turjanski, Fernando Asteasuain, Esteban Pavese, Charly López Pombo, Hernán Czemerinski, Pablo Factorovich... I'm sorry I can't name them all!

I had the opportunity to perform several research visits at the Talaris group (part of the Loria laboratory) in Nancy, France. I want to thank the people I met there, especially Guillaume Hoffmann, Yannick Parmentier and Sébastien Hinderer for very lively discussions and the people in the the "latin-american ghetto" of Nancy for making me feel like home: Luis Peñaranda, Leticia Basciano, Humberto Abdelnur, Alejandra Lorenzo, Lina Rojas Barahona, Luciana Benotti, Diego Patiño, Laura Pérez, Cristian Rosa, et al.

I want to thank also Isabel Méndez Díaz and Alejandro Ríos for their assistance with all the bureaucracy, and Miriam González for her invaluable help in getting the co-tutelle agreement going. Andrés Ferrari kindly designed the “Andran machine” for my defense.

Finally, I want to thank my friends and my family for their (blind) support, and especially Laura, for all her love, care, support, infinite patience and for the two tiny hands she knitted me so I could write at double speed.

Part I

Preliminaries

Chapter 1

Hybrid logics

The first modern approach to modal logics is commonly attributed to Lewis [1918]. In those early times, modal logics were all about *modes of truth*: necessity, obligation, etc. More than ninety years have passed since. Successful uses of modal logics can be found in an amazingly heterogeneous list of areas: from topology to software verification, from artificial intelligence to proof-theory, from linguistics to game-theory and the list goes on (see, e.g., part 4 of [Blackburn et al., 2006]).

Modal languages have been studied under an equally diverse number of interpretations: relational models and frames, neighborhood semantics, algebraic and co-algebraic settings, etc. The downside for the contemporary modal logician is that he can no longer give an answer that is both comprehensive and comprehensible when confronted by the layman with the recurrent question of what a modality is.

We won't attempt to give an answer to that question here either. In fact, we will restrict our attention to modal logics under *relational semantics*. Models in this setting are simply *relational structures* and therefore modal languages can be seen as languages for describing and reasoning about labeled graphs. This is, arguably, the interpretation that better suits computer scientists, who tend to see graphs everywhere.

Now, relational structures can be described with a variety of languages, including modal logics, first-order logics and even higher-order logics. The choice of a particular language for a particular application can be seen as a compromise between *expressiveness* (the properties we can express in the language) and *computational complexity* (the cost of performing inference tasks). Modal logics offer a particularly interesting balance of both. Moreover they can be used in a *modular* way: one can try to pick the modal operators that offer the best balance for each particular setting.

Hybrid logics augment modal logics with machinery for describing and reasoning about identity, which is a crucial in many settings. Before going into formal definitions, we will try to give an informal introduction to both modal and hybrid logics.

1.1 An example-driven introduction

One of the outstanding uses of logic in computer science is in the specification of software contracts. The most successful use of modal languages in this field is, arguably, in the specification of temporal properties for model-checking based verification of software and hardware [Clarke et al., 1999, Vardi, 2006]. In this section, however, we will use these languages –as a way to familiarize the reader with them– to describe recursive datastructures residing in the heap of some imperative Java-like programming language. These descriptions can be used, for example, as part of the declaration of a *class invariant* [Meyer, 1992]. Later in this example we will extend the range of these descriptions to *method contracts*.

Let us start by considering the code in Figure 1.1a, which describes a node type that can be used, for example, to build trees of arbitrary fan-out. Using only boolean connectives[1] we can state, for example, that "a node is a circle but not a red one":

$$\mathit{shape} = \text{Circle} \wedge \text{color} \neq \text{Red} \tag{1.1}$$

Now, to state that the same is true of every *immediate succesor* of the node (i.e., of every node contained in the succs set), we use a modal operator:

$$[\text{succs}](\mathit{shape} = \text{Circle} \wedge \text{color} \neq \text{Red}) \tag{1.2}$$

Observe that (1.2) is simply (1.1) with a prefixed [succs]. The intended interpretation is that for this formula to be true, (1.1) must be true at every immediate successor of the current node. Of course, if the node has no successor (i.e., if the set succs is empty), then (1.2) will be vacuously true. We can express that the node must have at least one successor using:

$$\neg[\text{succs}]\neg\top \tag{1.3}$$

where $\top$ is any tautology. Since [succs] has *universal* semantics ("every successor"), it comes as no surprise that an *existential* property ("some successor") is expressed this way. In fact, the existential dual of [succs] is typically written $\langle\text{succs}\rangle$; that is, for every formula φ, $\langle\text{succs}\rangle\varphi$ is shorthand for $\neg[\text{succs}]\neg\varphi$. Thus, (1.3) can be alternatively expressed as:

$$\langle\text{succs}\rangle\top \tag{1.4}$$

Operator [succs] is an example of a basic (relational) modality, that is, one that is interpreted universally using the labeled edges of an arbitrary

[1]We will assume throughout this section some suitable background theory that gives meaning to symbols such as $=$, $\neq$, Red, Circle, *shape*, etc.

```
class NodeRose {
  Color color;
  Shape shape;
  Set⟨NodeRose⟩ succs;
}
```

(a) Rose tree

```
class NodeBin {
  Bool hasMark;
  NodeBin? left;
  NodeBin? right;
}
```

(b) Binary tree

Figure 1.1: Data structure definitions in a Java-like language

graph. Certainly, every boolean tautology is a tautology in the language enriched with this modal operator, but now we also have new non-boolean ones. Consider, for example, the following one:

$$\begin{aligned}&[\text{succs}](\text{color} = \text{Red} \to [\text{succs}]\bot) \to \\ &\qquad \big([\text{succs}](\text{color} = \text{Red}) \to [\text{succs}][\text{succs}]\bot\big)\end{aligned} \tag{1.5}$$

To see that (1.5) is a valid formula, assume an arbitrary node of which you know that every (if any) successor satisfies $(\text{color} = \text{Red} \to [\text{succs}]\bot)$. If it also happens that every successor satisfies $\text{color} = \text{Red}$, then every successor satisfies both $(\text{color} = \text{Red} \to [\text{succs}]\bot)$ and $\text{color} = \text{Red}$, and from here we conclude that every successor satisfies also $[\text{succs}]\bot$.

Here is another example of valid formula:

$$[\text{succs}](\text{color} = \text{Red} \vee \text{color} \neq \text{Red}) \tag{1.6}$$

Since $(\text{color} = \text{Red} \vee \text{color} \neq \text{Red})$ is a (boolean) tautology, every successor must surely make it true.

But observe that in this argument we haven't used any other property of the formula $(\text{color} = \text{Red} \vee \text{color} \neq \text{Red})$ other of it being universally valid, so we can replace it with any other tautology. This justifies the following inference rule, known as the *Neccessitation rule.*

$$\textit{if } \varphi \textit{ is valid, then } [\textit{succs}]\varphi \textit{ is valid too.} \tag{1.7}$$

Similarly, we can generalize the analysis we did for (1.5) above and conclude that for any formulas φ and ψ, the following holds:

$$[\textit{succs}](\varphi \to \psi) \to [\textit{succs}]\varphi \to [\textit{succs}]\psi \textit{ is valid} \tag{1.8}$$

This is the so-called *axiom K* applied to modality [succs].

Interestingly axiom K and Necessitation, together, characterize the set of valid formulas of the basic modal logic. This is a decidable set (unlike,

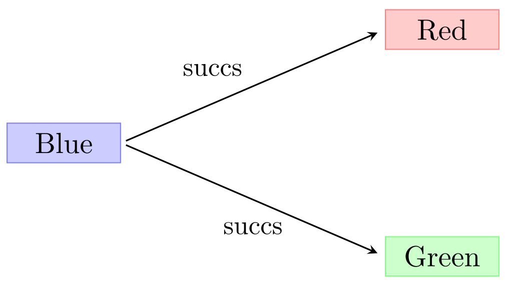

Figure 1.2: $\langle \text{succs} \rangle(\text{color} = \text{Red}) \rightarrow [\text{succs}](\text{color} = \text{Red})$ is not valid

for instance, the set of valid formulas of first-order logic) and membership in this set is a *PSPACE*-complete problem [Ladner, 1977].

Consider now the type NodeBin defined in Figure 1.1b. We use notation "NodeBin? *left*" to express that the field *left* may be empty (i.e., it may contain the *null* value). In other words, every instance of NodeBin may have at most one immediate *left*-successor and at most one immediate *right*-successor. Say we now want to express that "the right successor (if present) of the left succesor (if present) of the current node is marked"; this is very succinctly stated as:

$$[\,\mathit{left}\,][\mathit{right}]\mathit{hasMark} \tag{1.9}$$

Obviously enough, $[\,\mathit{left}\,]$ and $[\mathit{right}]$ are modal operators in the same sense [succs] is too: they satisfy axiom K and the Necessitation rule. They satisfy additional properties, though. For example, for every φ, the following are universally valid:

$$\langle\,\mathit{left}\,\rangle\varphi \rightarrow [\,\mathit{left}\,]\varphi \tag{1.10}$$

$$\langle\mathit{right}\rangle\varphi \rightarrow [\mathit{right}]\varphi \tag{1.11}$$

For the case of (1.10), if there exists some successor via *left* that satisfies φ, then this must be the only such successor, so it is true that every successor via *left* satisfies φ. On the other hand, Figure 1.2 shows this is not the case of [succs].

In fact, (1.10) and (1.11) are instances of axiom Alt_1, which together with axiom K and Necessitation characterizes the set of valid formulas when the relation used to interpret the operator is a partial function.

Instead of constraining the interpretation, one can also obtain other modal operators by adding new constructions to the language. The next two modalities we will introduce illustrate this. They are part of Propositional Dynamic Logic (PDL) [Harel et al., 2000].

To motivate the first one, observe that although it is easy to express a class invariant requiring that "every immediate successor of a NodeBin has to be marked":

$$[\mathit{left}]\mathit{hasMark} \wedge [\mathit{right}]\mathit{hasMark} \tag{1.12}$$

if we want to say instead that "every node at distance two has to be marked" things start to become verbose:

$$\begin{gathered}[\mathit{left}]([\mathit{left}]\mathit{hasMark} \wedge [\mathit{right}]\mathit{hasMark}) \quad \wedge \\ [\mathit{right}]([\mathit{left}]\mathit{hasMark} \wedge [\mathit{right}]\mathit{hasMark})\end{gathered} \tag{1.13}$$

And things get worse as we increase the depth of the nodes we want to talk about. This can be expressed very succinctly using PDL's *non-deterministic choice* modality constructor. For example, the following formula expresses that "every node at distance *four* is marked"

$$[\mathit{left} \cup \mathit{right}][\mathit{left} \cup \mathit{right}][\mathit{left} \cup \mathit{right}][\mathit{left} \cup \mathit{right}]\mathit{hasMark} \tag{1.14}$$

The relevant axiom in this case is: $[\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$, for $[\alpha]$ and $[\beta]$ modalities in our language. But we can go further. We can use PDL's *transitive-closure* modality constructor to express, for example, that "every node reachable in a finite number of steps has to be marked":

$$\mathit{hasMark} \wedge [(\mathit{left} \cup \mathit{right})^{+}]\mathit{hasMark} \tag{1.15}$$

Notice that we have used a modal language to write a property not expressible in first-order logic (namely, transitive closure). And we are staying decidable: validity for PDL is an *EXPTIME*-complete problem [Pratt, 1979, Fischer and Ladner, 1979].

It's time for a quick recap. We have seen that modal languages are well-suited for describing properties of graphs. Moreover, we have also seen that they are intrinsically *modular*: one can simply put together a bunch of modal operators and obtain a new logic. By adding appropriate modalities one can augment the expressive power of the resulting logic. But the additional expressive power often results in an increased computational complexity. In short, modal logics lend well to what has been called *logic engineering*: the ability to, given a concrete application, pick the language that expresses as much as one needs without *paying* in terms of complexity more than one can afford [Areces, 2000].

We will rely on our running example to also motivate and introduce hybrid logics. Consider the structures on Figure 1.3; call $\mathcal{M}$ the one depicted on (a) and $\mathcal{N}$ the one on (b). Pick any of the two marked nodes of $\mathcal{M}$, call it v and let v' be the marked node of $\mathcal{N}$. Using a very simple inductive argument one can show that for any formula φ of any of the languages we have considered so far, φ is true at v iff it is true at v'. Intuitively, v and v' are indistinguishable at the propositional level and because they have no successors, they trivially satisfy any formula of the form $[\alpha]\psi$. Now, call w the unmarked element of $\mathcal{M}$ and w' the unmarked element of $\mathcal{N}$; using

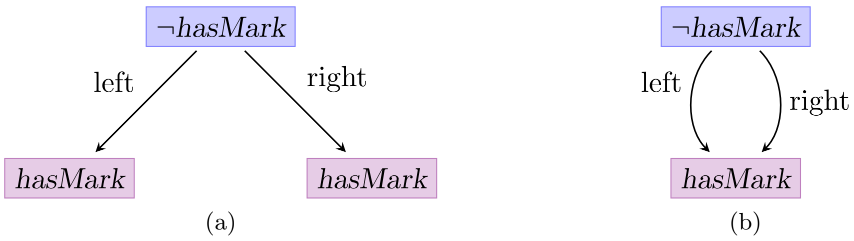

Figure 1.3: Two structures undistinguishable without hybrid operators.

another induction one can show that φ is true at w iff is is true at w'. Here, the important step is that (because of the first induction) the *left*-successor of w satisfies ψ iff the *left*-successor of w' satisfies ψ (and therefore, w satisfies $[\mathit{left}]\psi$ iff w' satisfies it too), etc[2].

Put in another way, even with the very expressive PDL operators, we have no way to state in the class invariant of a NodeBin that there must be no *aliasing*[3] between the immediate successors of a NodeBin.

Hybrid logics can be regarded as filling this expressive gap. These logics are built around a special sort of proposition symbol, that is required to be true at one and only one node of the model. These special symbols are called *nominals*. By simply adding nominals we obtain a richer logic, as illustrated by the following formula which is universally valid when i is a nominal:

$$\begin{aligned}(\langle \text{succs}\rangle(i \wedge \text{color} = \text{Red}) \wedge \langle \text{succs}\rangle(i \wedge \mathit{shape} = \text{Circle})) \rightarrow \\ \langle \text{succs}\rangle(\text{color} = \text{Red} \wedge \mathit{shape} = \text{Circle})\end{aligned} \tag{1.16}$$

Whenever the antecedent holds, we must have the only node *named* i as an immediate successor, but since this node must satisfy both color $=$ Red and *shape* $=$ Circle, the consequent must hold too.

Once we have nominals, we can add some interesting operators. Firstly, to every nominal i we can associate a modal operator $@_i$, that makes the evaluation of its subformula relative to the node named i. This is usually called the *satisfaction operator*. To continue developing our running example, observe that *program variables* make a fine example of nominals. For instance, in the following function declaration both a and b denote one and only one node each:

```
void doSomething(NodeRose a, NodeRose b)
```

[2]In fact, readers familiar with the concept of *bisimulation* will reckon that the structures of Figure 1.3a and 1.3b are bisimilar. It is well-known that modal formulas in the languages we considered so far are invariant under bisimulation and this constitutes a shorter proof of their indistinguishability.

[3]In computing, *aliasing* occurs when the same object or memory location can be accessed through different pointers or references.

Until now our example formulas were thought of as class invariants. In a class invariant there is no need to explicitly name the instance where it will be evaluated. But using the satisfaction operator we can start writing more general predicates, such as Hoare-style pre and postconditions or loop invariants. For instance, we can state that "a has no successors and b is red and has only red successors":

$$@_a[\text{succs}]\neg\top \wedge @_b(\text{color} = \text{Red} \wedge [\text{succs}](\text{color} = \text{Red})) \tag{1.17}$$

Moreover, we can also state that there is no aliasing between a and b:

$$@_a\neg b \tag{1.18}$$

That is, we are saying that the node named a is not also named b, which is only possible if a and b designate distinct nodes. Thus, the combination of nominals with the satisfaction operator is incorporating a notion of *equality* to the modal language.

It is fairly easy to express the requirement that "b is not a successor of itself":

$$@_b[\text{succs}]\neg b \tag{1.19}$$

but this is because we already have a name for b. What if we wanted to say also that "no successor of b is successor of itself"? Here enters the second hybrid operator we will introduce. Given a nominal i, the $\downarrow i$ operator names the current node with i for the rest of its subformula. The last statement is thus expressed as:

$$@_b[\text{succs}]{\downarrow}i.[\text{succs}]\neg i \tag{1.20}$$

This can be alternatively read as "let i be any successor of b, then i is not one of its own successors". We typically say that the occurrence of nominal i in (1.20) is *bound* by $\downarrow i$, and therefore, $\downarrow$ by itself is sometimes called a *binder* or *binding operator*.

Observe that using all the hybrid operators we can now write a formula that distinguishes the structures in Figure 1.3:

$${\downarrow}\mathit{root}.\langle\, \mathit{left}\,\rangle{\downarrow}\mathit{leftChild}.@_{\mathit{root}}\langle \mathit{right}\rangle\neg \mathit{leftChild} \tag{1.21}$$

To verify this, let again w and w' be the unmarked nodes of Figures 1.3a and 1.3b, respectively. We start by verifying that (1.21) is true at w. First, observe it starts requiring we assume w is denoted by nominal *root*. Next, it asks w to have some *left*-successor (which w does) and requires also ${\downarrow}\mathit{leftChild}.@_{\mathit{root}}\langle \mathit{right}\rangle\neg \mathit{leftChild}$ to be true at it, which we have to verify. This starts by assuming this *left*-successor is called *leftChild* and then moves us back to w (remember it was assumed to be called *root*) and requires of w that $\langle \mathit{right}\rangle\neg \mathit{leftChild}$ holds. But w indeed has a *right*-successor and it is a node distinct from the *left*-successor of w which was named *leftChild*. It is this last bit that fails in the case of w': the *right*-successor of w' is indeed the node that previously was named *leftChild*.

It comes as no surprise that this additional expressive power comes with a price: hybrid logics that include the $\downarrow$ binder have in general an undecidable validity problem. On the other hand, the satisfiability problem for the hybrid logic with nominals and the satisfaction operator remain *PSPACE*-complete [Areces et al., 1999].

Hopefully, the example in this section served to give an idea of what one can expect from modal languages and what is the role played by hybrid operators. Of course, we have barely started to scratch the surface. A thorough presentation of these and other modal languages, as well as some of their applications, can be found in [Blackburn et al., 2006].

1.2 All the due formalities

It is time to complement the intuitions we have developed so far with formal definitions. We will introduce here the basic concepts to be used throughout the thesis, although some of them will be later revised in Chapter 10.

1.2.1 Syntax and semantics

Let's begin by properly defining the hybrid language. We will usually work with a fixed signature $\mathcal{S}$, defined as the triple $\langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel} \rangle$, where Prop, Nom and Rel are infinite, enumerable, pairwise disjoint sets. Occasionally, we will need to work with more than one signature (e.g., an expanded one) and when that happens we will make them explicit.

Rel is the set of *relation symbols*; in the previous example succs, *left* and *right* were used as relation symbols but we will stick to the more abstract $r_1, r_2, r_3 \ldots$ (sometimes omitting the subscript) from now on, instead. Similarly, we won't use complex propositions like "color = Red" as elements of Prop but letters p, q (perhaps with subscripts). We reserve letters $i, j, k, \ldots$ for nominals in Nom. Finally, we will use letters $a, b, c \ldots$ to denote any atomic symbol, i.e., either a nominal or a proposition.

Definition 1.1 (Basic syntax)**.** The set of formulas of the hybrid logic $\mathcal{H}(@, \downarrow)$ (over signature $\mathcal{S}$) is defined as follows:

$$\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid [r]\varphi \mid @_i\varphi \mid \downarrow i.\varphi$$

We will also use all the usual abbreviations:

$$\begin{aligned}
\varphi \vee \psi &\overset{def}{=} \neg(\neg\varphi \wedge \neg\psi) \\
\varphi \rightarrow \psi &\overset{def}{=} \neg\varphi \vee \psi \\
\varphi \leftrightarrow \psi &\overset{def}{=} (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \\
\langle r \rangle\varphi &\overset{def}{=} \neg[r]\neg\varphi
\end{aligned}$$

The sublogics $\mathcal{H}(@)$ and $\mathcal{H}(\downarrow)$ are obtained by removing from $\mathcal{H}(@, \downarrow)$ every formula containing $\downarrow$ and $@$, respectively.

Operator $\downarrow$ is usually called a *binder* and, as such, induces the standard notions of *free* and *bound nominals*.

Definition 1.2 (Free and bound nominals)**.** The sets $\mathit{Free}(\varphi)$ and $\mathit{Bnd}(\varphi)$ of *free* and *bound* nominals occurring in φ are inductively defined as follows:

$$\begin{array}{rcl|rcl}
\mathit{Free}(p) & = & \emptyset & \mathit{Bnd}(p) & = & \emptyset \\
\mathit{Free}(i) & = & \{i\} & \mathit{Bnd}(i) & = & \emptyset \\
\mathit{Free}(\neg\psi) & = & \mathit{Free}(\psi) & \mathit{Bnd}(\neg\psi) & = & \mathit{Bnd}(\psi) \\
\mathit{Free}(\psi \wedge \chi) & = & \mathit{Free}(\psi) \cup \mathit{Free}(\chi) & \mathit{Bnd}(\psi \wedge \chi) & = & \mathit{Bnd}(\psi) \cup \mathit{Bnd}(\psi) \\
\mathit{Free}([r]\psi) & = & \mathit{Free}(\psi) & \mathit{Bnd}([r]\psi) & = & \mathit{Bnd}(\psi) \\
\mathit{Free}(@_i\psi) & = & \mathit{Free}(\psi) \cup \{i\} & \mathit{Bnd}(@_i\psi) & = & \mathit{Bnd}(\psi) \\
\mathit{Free}(\downarrow i.\psi) & = & \mathit{Free}(\psi) \setminus \{i\} & \mathit{Bnd}(\downarrow i.\psi) & = & \mathit{Bnd}(\psi) \cup \{i\}
\end{array}$$

Moreover, we say i *is free in* φ whenever $i \in \mathit{Free}(\varphi)$ and that i *is bound in* φ whenever $i \in \mathit{Bnd}(\varphi)$.

In many presentations of hybrid logics (e.g., [Areces and ten Cate, 2006]) $\downarrow$ does not bind nominals but another sort of symbols, typically called *state variables*. We have opted to drop state variables from our definitions basically to avoid redundancies both in syntax and semantics. There is a small price we have to pay for this, though. Proof methods for $\mathcal{H}(@, \downarrow)$ usually rely on *substitutions* of variables by nominals. For example, the following is an axiom-scheme of a Hilbert-style system for $\mathcal{H}(@, \downarrow)$ (cf. Chapter 4):

$$\vdash @_i(\downarrow x.\varphi \leftrightarrow \varphi(x/i)) \tag{1.22}$$

where x is a state variable and $\varphi(x/i)$ is the substitution inside φ of the free occurrences of x by i. But by dropping state variables we open the door to the possibility of the accidental capture of nominals during replacement. For example, if we simply replaced (1.22) by the following:

$$\vdash @_i(\downarrow j.\varphi \leftrightarrow \varphi(j/i)) \tag{1.23}$$

we would get, for $\varphi = \langle r\rangle{\downarrow} i.\neg j$, the spurious axiom

$$\vdash @_i(\downarrow j.\langle r\rangle{\downarrow} i.\neg j \leftrightarrow \langle r\rangle{\downarrow} i.\neg i) \tag{1.24}$$

which is not valid. The workaround for this problem will be simply to adopt a convention in the spirit of the *Variable Convention* for the λ-calculus used by Barendregt [1984] that restores in practice the distinction between *nominal* and *variable* but without all the associated formal burden.

Convention 1. We will always assume that if φ is a well-formed formula of $\mathcal{H}(@, \downarrow)$, then $\mathit{Free}(\varphi) \cap \mathit{Bnd}(\varphi) = \emptyset$.

This convention is not too restrictive since one can always rename appropriately the bound nominals. It is also convenient since every subformula of a well-formed formula will conform to the convention by construction. Finally, because of this convention, it will be enough to use a naive notion of substitution:

Definition 1.3 ($\varphi(i/j)$)**.** Let φ be a $\mathcal{H}(@,\downarrow)$ formula and $j \notin \mathit{Bnd}(\varphi)$. We define the *uniform substitution of free occurrences of i by j*, $\varphi(i/j)$, inductively as follows:

$$\begin{aligned}
i(i/j) &\overset{def}{=} j \\
a(i/j) &\overset{def}{=} a, \text{ if } a \neq i \\
(\neg\psi)(i/j) &\overset{def}{=} \neg(\varphi(i/j)) \\
(\psi \wedge \chi)(i/j) &\overset{def}{=} \psi(i/j) \wedge \chi(i/j) \\
([r]\psi)(i/j) &\overset{def}{=} [r](\psi(i/j)) \\
(@_i\psi)(i/j) &\overset{def}{=} @_j(\psi(i/j)) \\
(@_k\psi)(i/j) &\overset{def}{=} @_k(\psi(i/j)), \text{ if } k \neq i \\
(\downarrow i.\psi)(i/j) &\overset{def}{=} \downarrow i.\psi \\
(\downarrow k.\psi)(i/j) &\overset{def}{=} \downarrow k.(\psi(i/j)), \text{ if } k \neq i
\end{aligned}$$

Notice condition $j \notin \mathit{Bnd}(\varphi)$ in Definition 1.3 above. Its presence guarantees that no substitution may break the convention by replacing a free nominal by one that is also bound.

Having formally defined the hybrid language, we should warn upfront that this is not the only characterization of $\mathcal{H}(@,\downarrow)$ we are going to use. In Parts II and III we will assume formulas are in negation normal form (nnf). In this form, negations occur only in front of atoms. We will provide an inductive definition of nnf-formulas to allow for proofs by induction.

Definition 1.4 (Basic syntax (nnf))**.** The set of formulas in negation normal form of the hybrid logic $\mathcal{H}(@,\downarrow)$ (over signature $\mathcal{S}$) is defined as follows:

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [r]\varphi \mid \langle r\rangle\varphi \mid @_i\varphi \mid \downarrow i.\varphi$$

Of course, there is a linear translation *nnf* such that, for all φ, $\mathit{nnf}(\varphi)$ is in negation normal form and equivalent to φ. It is shown in Figure 1.4. In any case, it shall always be clear what language we are working in.

Time to move to semantics. As usual, we need a suitable notion of interpretation for the symbols in the signature.

Definition 1.5 (Models)**.** A *hybrid model* is a tuple $\langle W, R, V, g\rangle$ where W is a non-empty set, $R : \mathsf{Rel} \to 2^{W\times W}$, $V : \mathsf{Prop} \to 2^{W}$, $g : \mathsf{Nom} \to W$.

$$
\begin{array}{lll|lll}
nnf(a) & \overset{def}{=} & a & nnf^{-}(a) & \overset{def}{=} & \neg a \\
nnf(\neg\psi) & \overset{def}{=} & nnf^{-}(\psi) & nnf^{-}(\neg\psi) & \overset{def}{=} & nnf(\psi) \\
nnf(\psi_1 \wedge \psi_2) & \overset{def}{=} & nnf(\psi_1) \wedge nnf(\psi_2) & nnf^{-}(\psi_1 \wedge \psi_2) & \overset{def}{=} & nnf^{-}(\psi_1) \vee nnf^{-}(\psi_2) \\
nnf([r]\psi) & \overset{def}{=} & [r]nnf(\psi) & nnf^{-}([r]\psi) & \overset{def}{=} & \langle r\rangle nnf^{-}(\psi) \\
nnf(@_i\psi) & \overset{def}{=} & @_i nnf(\psi) & nnf^{-}(@_i\psi) & \overset{def}{=} & @_i nnf^{-}(\psi) \\
nnf(\downarrow i.\psi) & \overset{def}{=} & \downarrow i.nnf(\psi) & nnf^{-}(\downarrow i.\psi) & \overset{def}{=} & \downarrow i.nnf^{-}(\psi)
\end{array}
$$

Figure 1.4: The *nnf* transformation.

A model $\langle W, R, V, g\rangle$ can be seen as a directed graph with the edges labeled by relation symbols. Function g assigns a node of the graph to each nominal, and since type $\mathsf{Prop} \to 2^W$ is isomorphic to $W \to 2^{\mathsf{Prop}}$, we can view V as decorating each node of the graph with a propositional valuation.

If $\mathcal{M} = \langle W, R, V, g\rangle$ we say that W is the *domain of* $\mathcal{M}$ and sometimes denote it by $|\mathcal{M}|$. Following traditional practices, we may refer to elements of W as *nodes*, *states* or *worlds*.

Definition 1.6 ($\mathcal{M}_i^w$)**.** Given a model $\mathcal{M} = \langle W, R, V, g\rangle$, an element $w \in W$ and a nominal i, we define the model $\mathcal{M}_i^w$ as $\langle W, R, V, g'\rangle$ where g' is identical to g except perhaps for i for which $g'(i) = w$.

Since we've given two alternative characterizations of the set of hybrid formulas, we will formally define the semantics of the language for all the operators involved.

Definition 1.7 (Semantics)**.** Given a hybrid model $\mathcal{M} = \langle W, R, V, g\rangle$ and an element $w \in W$, the satisfiability relation $\mathcal{M}, w \models \varphi$ (read "model $\mathcal{M}$ satisfies formula φ at state w") is defined as follows:

$$
\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p) \\
\mathcal{M}, w \models i & \text{iff} & w = g(i) \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi_1 \wedge \varphi_2 & \text{iff} & \mathcal{M}, w \models \varphi_1 \text{ and } \mathcal{M}, w \models \varphi_2 \\
\mathcal{M}, w \models \varphi_1 \vee \varphi_2 & \text{iff} & \mathcal{M}, w \models \varphi_1 \text{ or } \mathcal{M}, w \models \varphi_2 \\
\mathcal{M}, w \models [r]\varphi & \text{iff} & (w, v) \in R(r) \text{ implies } \mathcal{M}, v \models \varphi, \text{ for every } v \in W \\
\mathcal{M}, w \models \langle r\rangle\varphi & \text{iff} & (w, v) \in R(r) \text{ and } \mathcal{M}, v \models \varphi, \text{ for some } v \in W \\
\mathcal{M}, w \models @_i\varphi & \text{iff} & \mathcal{M}, g(i) \models \varphi \\
\mathcal{M}, w \models \downarrow i.\varphi & \text{iff} & \mathcal{M}_i^w, w \models \varphi.
\end{array}
$$

We write $\mathcal{M} \models \varphi$ whenever $\mathcal{M}, w \models \varphi$ for every $w \in |\mathcal{M}|$.

From this definition it is clear that the $@_i$ operator moves the point of evaluation to the state named i and therefore $\mathcal{M}, w \models @_i\varphi$ iff $\mathcal{M} \models @_i\varphi$. Formulas whose outermost operator is an @ will play an important role in this thesis and will be called @-formulas. From all @-formulas, those of the form $@_ij$ will be called *equalities*; this is because for $\mathcal{M} = \langle W, R, V, g\rangle$, $\mathcal{M} \models @_ij$ iff $g(i) = g(j)$.

We say φ and ψ are *equivalent*, denoted $\varphi \equiv \psi$ whenever, $\models \varphi \leftrightarrow \psi$. Observe that $\neg @_i\varphi \equiv @_i\neg\varphi$ and $\neg{\downarrow}i.\varphi \equiv {\downarrow}i.\neg\varphi$. That is, they are *self-dual* operators.

Even though this thesis is about hybrid logics, we will often discuss the classic modal case too, typically when exploring in the hybrid setting ideas that have been used for the basic modal logic.

A modal signatures $\mathcal{S}$ is simply a tuple $\langle \mathsf{Prop}, \mathsf{Rel}\rangle$. A hybrid signature may be seen as a modal signature by simply ignoring Nom.

Definition 1.8 (Basic modal logic – syntax)**.** The set of formulas of the basic modal logic $\mathcal{ML}$ (over signature $\mathcal{S}$) is defined as follows:

$$\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid [r]\varphi$$

We will also use all the usual abbreviations (cf. Definition 1.1).

Models of the basic modal logic are usually called *Kripke models*. Having presented hybrid models first, Kripke models can be view as hybrid models without an interpretation for nominals.

Definition 1.9 (Kripke models)**.** A *Kripke model* is a tuple $\langle W, R, V\rangle$ where W is a non-empty set, $R : \mathsf{Rel} \to 2^{W\times W}$ and $V : \mathsf{Prop} \to 2^W$.

Sometimes it is convenient to consider *pointed Kripke models*, which is simply a pair $\langle \mathcal{M}, w\rangle$, where $\mathcal{M}$ is a Kripke model and w an element of the domain of $\mathcal{M}$. We will come back to pointed models in Part IV.

Definition 1.10 (Basic modal logic – semantics)**.** Given a Kripke model $\mathcal{M} = \langle W, R, V\rangle$ and an element $w \in W$, the satisfiability relation $\mathcal{M}, w \models \varphi$ (read "model $\mathcal{M}$ satisfies formula φ at state w") is defined as follows:

$$\begin{array}{lll}
\mathcal{M}, w \models_K p & \text{iff} & w \in V(p) \\
\mathcal{M}, w \models_K \neg\varphi & \text{iff} & \mathcal{M}, w \not\models_K \varphi \\
\mathcal{M}, w \models_K \varphi_1 \wedge \varphi_2 & \text{iff} & \mathcal{M}, w \models_K \varphi_1 \text{ and } \mathcal{M}, w \models_K \varphi_2 \\
\mathcal{M}, w \models_K [r]\varphi & \text{iff} & (w, v) \in R(r) \text{ implies } \mathcal{M}, v \models_K \varphi, \forall v \in W.
\end{array}$$

We write $\mathcal{M} \models_K \varphi$ whenever $\mathcal{M}, w \models_K \varphi$ for every $w \in |\mathcal{M}|$. Finally, we define $\mathsf{K} = \{\varphi \mid \mathcal{M} \models_K \varphi \text{for all } \mathcal{M}\}$.

In most logics, a formula is typically said to be *satisfiable* when there exists some interpretation that makes it true, and *valid* when every interpretation makes it true. For example, in the case of $\mathcal{H}(@, \downarrow)$, we say φ is satisfiable whenever there exists $\mathcal{M}$ such that $\mathcal{M}, w \models \varphi$ for some $w \in |\mathcal{M}|$ and that it is valid if $\mathcal{M}, w \models \varphi$ for every $\mathcal{M}$ and every $w \in |\mathcal{M}|$.

The *satisfiability problem* of a logic is that of determining, given a formula φ, if φ is satisfiable or not. Similarly, the *validity problem* is that of deciding if a formula φ is valid. Validity and satisfiability are dual concepts: φ is valid iff $\neg\varphi$ is unsatisfiable. Therefore, any (deterministic) algorithm for the satisfiability problem can be trivially turned into one for the validity problem and vice versa.

The satisfiability problem for $\mathcal{H}(@)$ is *PSPACE*-complete, just like the one for $\mathcal{ML}$. On the other hand, the satisfiability problem for $\mathcal{H}(\downarrow)$ is undecidable (therefore, satisfiability for $\mathcal{H}(@, \downarrow)$ is undecidable too), but validity (or unsatisfiability) is semi-decidable [Areces et al., 1999].

As is usual when discussing complexity in the context of automated reasoning, it is worth noticing that one should not be necessarily discouraged by this hard complexities. After all, complexity-theorists refer to *NP*-complete problems as *intractable* while advanced satisfiability solvers are used nowadays on problems in this complexity class containing tens of thousands of variables. Assume $P \neq NP$ for a moment; *NP*-completeness of the propositional satisfiability problem then says that for every algorithm there exist pathological formulas for which it will require non-polynomial time. But it does not follow that these pathological formulas need to occur frequently in practice.

1.2.2 Bisimulations and modal invariance

The notion of *bisimulation* is central in modal model theory, but also in areas as diverse as concurrency theory or non-well founded set theory, in which they were independently discovered. In modal logics, bisimulations were introduced by van Benthem [1976]. See [Sangiorgi, 2009] for a historical review.

Definition 1.11 (Bisimulations for $\mathcal{ML}$)**.** Let $\mathcal{M} = \langle W, R, V \rangle$ and $\mathcal{M}' = \langle W', R', V' \rangle$ be two Kripke models. A *bisimulation* between $\mathcal{M}$ and $\mathcal{M}'$ is a non-empty binary relation $Z \subseteq W \times W'$ such that, if $(w, w') \in Z$, then:

(atom) $w \in V(p)$ iff $w' \in V'(p)$, for all $p \in \mathsf{Prop}$.

(zig) if $(w, v) \in R(r)$, then $(w', v') \in R'(r)$ and $(v, v') \in Z$, for some v'.

(zag) if $(w', v') \in R'(r)$, then $(w, v) \in R(r)$ and $(v, v') \in Z$, for some v.

When there exists a bisimulation Z between $\mathcal{M}$ and $\mathcal{M}'$ such that $(w, w') \in Z$, we say that *w and w' are bisimilar*, notated $\mathcal{M}, w \underline{\leftrightarrow} \mathcal{M}', w'$.

Theorem 1.1 (Invariance under bisimulations for $\mathcal{ML}$)**.** *Let $\mathcal{M}$ and $\mathcal{N}$ be two Kripke models. If $\mathcal{M}, w \underline{\leftrightarrow} \mathcal{N}, v$, then $\mathcal{M}, w \models_K \varphi$ iff $\mathcal{N}, v \models_K \varphi$, for all formula φ of $\mathcal{ML}$.*

Stronger links between modal equivalence (i.e., agreement on all modal formulas) and bisimulations can be found. Hennessy and Milner [1985] show that for models that are *finitely branching* (i.e., models where every node has finitely many successors) modal equivalence implies existence of a bisimulation (this does not hold in the general case). Moreover, van Benthem [1976] proved that every first-order logic formula that is invariant under bisimulations is equivalent to some modal formula (this is known as the van Benthem Characterization Theorem; we will delay a formal presentation of the links between first-order logic and modal logics until Part II of this thesis).

One can use bisimulations to prove that modal logics are invariant under certain operations on models. Classical operations for the basic modal case include *bounded morphic images*, *generated submodels* and *disjoint unions*. We formally define the latter next as we will need it in Chapter 6 (generated submodels for the hybrid case are presented in Definition 1.14).

Definition 1.12. Given two Kripke models $\mathcal{M}_1 = \langle W_1, R_1, V_1 \rangle$ and $\mathcal{M}_2 = \langle W_2, R_2, V_2 \rangle$, we define $\mathcal{M}_1 \uplus \mathcal{M}_2$, the disjoint union of $\mathcal{M}_1$ and $\mathcal{M}_2$, as the model $\langle W, R, V \rangle$, where:

$$\begin{aligned} W &\overset{def}{=} \{(0, w) \mid w \in W_1\} \cup \{(1, w) \mid w \in W_2\} \\ R(r) &\overset{def}{=} \{((0, w), (0, v)) \mid (w, v) \in R_1(r)\} \cup \{((1, w), (1, v)) \mid (w, v) \in R_2(r)\} \\ V(p) &\overset{def}{=} \{(0, w) \mid w \in V_1(p)\} \cup \{(1, w) \mid w \in V_2(p)\}. \end{aligned}$$

Intuitively, 0 and 1 are used to index the elements of the domain of the disjoint union according to which set they originated in. Now, consider the following binary relations:

$$Z_0 = \{(w, (0, w)) \mid w \in |\mathcal{M}_1|\} \tag{1.25}$$

$$Z_1 = \{(w, (1, w)) \mid w \in |\mathcal{M}_2|\} \tag{1.26}$$

Clearly, Z_1 is a bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_1 \uplus \mathcal{M}_2$, while Z_2 is a bisimulation between $\mathcal{M}_2$ and $\mathcal{M}_1 \uplus \mathcal{M}_2$. This justifies the following result.

Proposition 1.1 (Invariance under disjoint unions)**.** *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two Kripke models. Then, for all $w_1 \in |\mathcal{M}_1|$, $w_2 \in |\mathcal{M}_2|$ and all formula φ of $\mathcal{ML}$, we have:*

1. *$\mathcal{M}_1, w_1 \models_K \varphi$ iff $\mathcal{M}_1 \uplus \mathcal{M}_2, (0, w_1) \models_K \varphi$,*
2. *$\mathcal{M}_2, w_2 \models_K \varphi$ iff $\mathcal{M}_1 \uplus \mathcal{M}_2, (1, w_2) \models_K \varphi$.*

The link between bisimulations and modal logics goes beyond the basic language. Variations of the basic notion of bisimulation given in Definition 1.11 are typically used to characterize the expressive power of more expressive modal logics (but we will come back to this subject in Part IV). To illustrate this, we consider next the adequate notion of bisimulation for $\mathcal{H}(@)$ (the notion for $\mathcal{H}(@, \downarrow)$ requires heavier machinery, details can be found in [Areces, 2000]).

Definition 1.13 (Bisimulations for $\mathcal{H}(@)$)**.** Let $\mathcal{M} = \langle W, R, V, g\rangle$ and $\mathcal{M}' = \langle W', R', V', g'\rangle$ be two hybrid models. A *bisimulation* between $\mathcal{M}$ and $\mathcal{M}'$ is a relation $Z \subseteq W \times W'$ that it is a modal bisimulation (cf. Definition 1.11) between $\langle W, R, V\rangle$ and $\langle W', R', V'\rangle$ and additionally satisfies:

(atom') if $(w, w') \in Z$, then $g(i) = w$ iff $g'(i) = w'$, for all $i \in \mathsf{Nom}$.

(nom) Z extends $\{(g(i), g'(i)) \mid \text{for all } i \in \mathsf{Nom}\}$.

A bisimulation for $\mathcal{H}(@)$ is simply a bisimulation for $\mathcal{ML}$ with the additional condition that every named world must be in the bisimulation. Intuitively, this accounts for the fact that using @ one can *jump* to any element of the domain named by a nominal.

Theorem 1.2 (Invariance under bisimulations for $\mathcal{H}(@)$)**.** *Let $\mathcal{M}$ and $\mathcal{N}$ be two hybrid models. If $\mathcal{M}, w \underline{\leftrightarrow} \mathcal{N}, v$, then $\mathcal{M}, w \models \varphi$ iff $\mathcal{N}, v \models \varphi$, for all formula φ of $\mathcal{H}(@)$.*

As an example of a formula preserving operation on hybrid models, we define next the equivalent of generated submodels for the hybrid case; we will use it in Chapter 6.

Definition 1.14 (Generated submodels)**.** Let $\mathcal{M} = \langle W, R, V, g\rangle$ and $\mathcal{M}' = \langle W', R', V', g'\rangle$ be two hybrid models. We say that $\mathcal{M}'$ *is a generated submodel of* $\mathcal{M}$ whenever:

1. $W' \subseteq W$,
2. $R'(r)$ is the restriction of $R(r)$ to W' (i.e., $R'(r) = R(r) \cap (W' \times W')$),
3. V' is the restriction of V to W' (i.e., $V' = V \cap (W' \times W')$),
4. $g' = g$, which implies that W' contains the image of g, and
5. if $w \in W'$ and $(w, v) \in R(r)$ for some $r \in \mathsf{Rel}$, then $v \in W'$.

Observe that from Theorem 1.2 and the fact that the identity on the domain of $\mathcal{M}'$, a generated submodel of $\mathcal{M}$, is a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$, one concludes that $\mathcal{H}(@)$-formulas are invariant under generated submodels. Areces et al. [2001a] show that this also holds in the case of $\mathcal{H}(@, \downarrow)$ (in fact, they prove that if a first-order logic formula is invariant under generated submodels, it is equivalent to a formula of $\mathcal{H}(@, \downarrow)$).

Proposition 1.2 (Invariance under generated submodels)**.** *If $\mathcal{M}'$ is a generated submodel of $\mathcal{M}$, then $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}', w \models \varphi$, for all $w \in |\mathcal{M}'|$ and every formula φ of $\mathcal{H}(@, \downarrow)$.*

The classical notion of generated submodel for $\mathcal{ML}$ simply ignores the condition on the interpretation of nominals. The invariance result is analogous.

Bisimulations characterize the expressive power of modal logics in a way that reminds of *partial isomorphisms* in the case of first-order logic. By considering *bisimulations up to a finite number of steps* one obtains a modal equivalent of the classical Ehrenfeucht-Fraïssé characterization.

Definition 1.15 (k-bisimulations for $\mathcal{H}(@)$)**.** Given two hybrid models $\mathcal{M} = \langle W, V, R, g\rangle$ and $\mathcal{M}' = \langle W', V', R', g'\rangle$ and two elements $w \in W$, $w' \in W'$, we say that *w and w' are k-bisimilar* (notation, $\mathcal{M}, w \underline{\leftrightarrow}_k \mathcal{M}', w'$) if there exists a sequence of binary relations $W \times W' \supseteq Z_0 \supseteq Z_1 \supseteq \cdots \supseteq Z_k$ such that

1. $(w, w') \in Z_k$,
2. $\{(g(i), g'(i)) \mid i \in \mathsf{Nom}\} \subseteq Z_k$,
3. if $(v, v') \in Z_0$, then $v \in V(p)$ iff $v' \in V'(p)$, for all $p \in \mathsf{Prop}$, and
4. if $(v, v') \in Z_{i+1}$, then
   (a) if $(v, u) \in R(m)$, $(u, u') \in Z_i$ and $(v', u') \in R'(m)$, for some u',
   (b) if $(v', u') \in R'(m)$, $(u, u') \in Z_i$ and $(v, u) \in R(m)$, for some u.

Where in first-order logic one uses the *quantifier depth* of a formula, in modal settings we use the *modal depth* (sometimes called *degree*) of a formula instead.

Definition 1.16 (Modal depth)**.** The *modal depth* of a formula φ of $\mathcal{H}(@, \downarrow)$, (notation, $md(\varphi)$) is defined as:

$$\begin{aligned} md(p) &= 0 \\ md(i) &= 0 \\ md(\neg\varphi) &= md(\varphi) \\ md(\varphi \wedge \psi) &= \max\{md(\varphi), md(\psi)\} \\ md([m]\varphi) &= md(\varphi) + 1 \\ md(@_i\varphi) &= md(\varphi) + 1 \\ md(\downarrow i.\varphi) &= md(\varphi) + 1 \end{aligned}$$

First, notice that although the next result will use it only for formulas of $\mathcal{H}(@)$, md is defined for the language $\mathcal{H}(@, \downarrow)$. Second, observe that in terms of counting modal nesting, $@_i$ and $\downarrow i$ are treated as regular (relational) modal operators (but we will come back to this subject in Part IV).

Theorem 1.3 (Invariance under k-bisimulations for $\mathcal{H}(@)$)**.** *Let $\mathcal{M}$ and $\mathcal{N}$ be two hybrid models. If $\mathcal{M}, w \underline{\leftrightarrow}_k \mathcal{N}, v$, then $\mathcal{M}, w \models \varphi$ iff $\mathcal{N}, v \models \varphi$, for all formula φ of $\mathcal{H}(@)$ such that $md(\varphi) \leq k$.*

1.2.3 Models and frames

If we drop the valuation V from a Kripke model $\langle W, R, V\rangle$ we obtain what is usually called a *frame* $\langle W, R\rangle$. Alternatively, a Kripke model $\langle W, R, V\rangle$ can be seen as a frame $\langle W, R\rangle$ plus a valuation V. There is a long-standing tradition of studying modal logics from the perspective of frames. One of the key concept here is that of *frame definability*.

Definition 1.17 (Frame definability)**.** Let C be a class of frames and φ a formula of $\mathcal{ML}$. We say that φ *defines* C whenever for every $\mathcal{F} = \langle W, R\rangle$, $\mathcal{F}$ belongs to $\mathcal{C}$ iff $\langle W, R, V\rangle \models_K \varphi$, for every valuation V.

Figure 1.5 shows some well-known examples of frame-defining formulas. The formula known as 5, for example, characterizes the class of frames whose relation r is *euclidean*, that is those that satisfy the first-order condition:

$$\forall xyz.((r(x,y) \wedge r(x,z)) \rightarrow r(y,z)). \tag{1.27}$$

We met *Alt₁* already, in Section 1.1; the first-order condition of the class of frames it defines is:

$$\forall xyz.((r(x,y) \wedge r(x,z)) \rightarrow y = z). \tag{1.28}$$

Now, observe the frame condition imposed by *Löb's formula*. Using a routine compactness argument, one can show that the class of frames defined by this formula is not elementary. The second-order quantification hidden in Definition 1.17 is responsible for this.

However, despite its second-order expressive power, there are elementary classes of frames that are not definable using a basic modal formula. Goldblatt and Thomason [1975] gave a precise characterization of the elementary frames that are modally definable.

Theorem 1.4 (Goldblatt-Thomason Theorem)**.** *A first-order definable class of frames is modally definable if and only if it is closed under taking bounded*

(T)	$[r]p \rightarrow p$	r is reflexive
(4)	$[r]p \rightarrow [r][r]p$	r is transitive
(5)	$\langle r\rangle[r]p \rightarrow [r]p$	r is euclidean
(Alt_1)	$\langle r\rangle p \rightarrow [p]$	r is a partial function
$(Löb)$	$[r]([r]p \rightarrow p) \rightarrow [r]p$	r is transitive and r^{-1} is well-founded

Figure 1.5: Examples of modal frame definability.

morphic images, generated subframes, disjoint unions, and reflects ultrafilter extensions.

Of the conditions mentioned in Theorem 1.4, we have presented so far only *disjoint unions* and *generated subframes*[4], the remaining ones can be found in any textbook on modal logics (e.g., [Blackburn et al., 2002]).

We can get an idea of what kind of elementary classes are not modally definable by looking for frame classes that are not closed under either disjoint unions or generated subframes. For the first-case, consider the class of frames where relation r is the total relation, that is, it satisfies:

$$\forall x \forall y . r(x,y) \tag{1.29}$$

This class is clearly not closed under disjoint unions and therefore is not modally definable. For the case of generated subframes, consider the class of frames such that every node is *reached* via relation r; or, in first-order terms:

$$\forall x \exists y . r(y,x) \tag{1.30}$$

Now, frame $\mathcal{F}_1 = \langle \mathbb{Z}, R_1 \rangle$ where $R_1(r) = \{(w,v) \mid w < v\}$ satisfies condition (1.30) while frame $\mathcal{F}_2 = \langle \mathbb{N}, R_2 \rangle$ where $R_2(r) = \{(w,v) \mid w < v\}$ is a generated subframe of $\mathcal{F}_1$ but does not satisfy (1.30). Therefore, the class is not closed under generated subframes and is not definable by a modal formula.

What about definability using hybrid formulas? Definition 1.17 is trivially extended to the hybrid case: a formula φ in $\mathcal{H}(@, \downarrow)$ defines a class C if a frame $\mathcal{F} = \langle W, R \rangle$ is in C iff $\langle W, R, V, g \rangle \models \varphi$, for all V and all g. Interestingly, the frame class given by (1.29) is defined by the hybrid formula $\langle r \rangle i$. However, frames that are not closed by generated subframes are not definable by hybrid formulas either. A precise characterization of frame definability in the hybrid case is given by ten Cate [2005].

Let $\mathcal{C}$ be a fixed class of models; the *satisfiability problem with respect to* $\mathcal{C}$ consists in determining if there exists a model belonging to $\mathcal{C}$ that satisfies a given formula. In general, satisfiability (that is, with respect to the class of all models) and satisfiability with respect to an arbitrary class are independent problems.

It is well-known that $\mathcal{H}(@, \downarrow)$ is a *conservative reduction class*, that is, there exists a recursive function τ that maps arbitrary first-order logic formulas to $\mathcal{H}(@, \downarrow)$ such that for all φ, $\tau(\varphi)$ is satisfiable iff φ is, and $\tau(\varphi)$ has a finite model iff φ has. ten Cate [2005] provides a proof of this property from which it follows that for any elementary class of models $\mathcal{C}$, there exists a recursive mapping $f_{\mathcal{C}}$ such that, for any formula φ of $\mathcal{H}(@, \downarrow)$, φ is

[4]They were presented as operations on models, they are turned into operations on frames by ignoring the valuation.

satisfiable with respect to $\mathcal{C}$ iff $f_{\mathcal{C}}(\varphi)$ is satisfiable with respect to the class of all models.

While this means that, in theory, satisfiability of a formula of any sublogic of $\mathcal{H}(@, \downarrow)$ with respect to a first-order definable class $\mathcal{C}$ may be reduced to general $\mathcal{H}(@, \downarrow)$-satisfiability, in practice this will be far from optimal. Observe, for example, that $\mathcal{H}(\downarrow)$-satisfiability with respect to the class of models where all the relations are transitive is a decidable problem (in fact, *NEXPTIME*-complete [Mundhenk et al., 2005]).

In any case, the main concern of this thesis will be satisfiability over the class of all models. In the following chapter we review the basic resolution-based techniques we will consider.

Chapter 2

Resolution

Having introduced hybrid logics, it is time to talk about the second theme of this thesis: resolution-based automated reasoning techniques. There is a large diversity of such techniques a detailed survey is out of our scope. We will only present, to make this thesis self-contained, the methods (some standard, some non-standard) that we will revisit in the remaining chapters. For reference on classical automated deduction techniques, the reader is referred to [Robinson and Voronkov, 2001]. Horrocks et al. [2006] survey reasoning techniques for modal logics.

2.1 Classical resolution and paramodulation

The resolution method for first-order logic was introduced by Robinson [1965] as a reasoning principle suitable for mechanic computation. Its main advantage over other early theorem proving methods is that unification, as a selection mechanism for inferences, provides an effective way of interleaving instantiation and refutation. Most modern automatic theorem provers for first-order logic are saturation-based and implement, at their core, variations of the original resolution rule. Throughout this section we will present the calculus of ordered resolution with selection, following the presentation by Bachmair and Ganzinger [2001].

The resolution method works on quantifier-free first-order formulas in clausal form. A formula in this form is a conjunction of clauses, where each clause is a disjunction of first-order literals. Implicitly, variables are universally quantified. Using some form of skolemization plus structural transformation rules, it is possible to put any formula in clausal form in a satisfiability preserving way requiring only polynomial time. For details on these transformations, refer to [Nonnengart and Weidenbach, 2001].

The inference rules of the ordered resolution calculus, denoted here as $\mathsf{R}^{\succ}_{S}$, are shown in Figure 2.1. These rules are parameterized by an *admissible ordering* $\succ$ and a selection function S. Essentially, an admissible ordering

$$\textsf{Resolution}\quad \frac{C \vee \neg A_1 \quad A_2 \vee D}{(C \vee D)\sigma} \qquad \textsf{Positive factoring}\quad \frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$

Global conditions

i. σ is the most general unifier of atoms A_1 and A_2.

ii. $\neg A_1 \in S(C)$ or, else, $S(C) = \emptyset$ and $\neg A_1\sigma$ is $\succ$-maximal wrt $C\sigma$.

iii. $S(D) = \emptyset$ and $A_2\sigma$ is strictly $\succ$-maximal wrt $D\sigma$.

Figure 2.1: Inference rules of $\mathsf{R}_S^{\succ}$

(on clauses) is the *multiset extension*[1] of an admissible ordering on literals, that is, a well-founded ordering that is total on the ground level and such that $\neg A \succ A$, for every atom A. Admissible orderings are lifted to non-ground clauses by stipulating $C \succ D$ iff $C\sigma \succ D\sigma$ for every ground substitution σ. A selection function S assigns to each clause a possibly empty set of occurrences of negative literals.

The resolution rule can be seen as a combination of variable instantiation with modus-ponens. To illustrate this, consider the following instance of the rule, where we resolve on P_1:

$$\frac{\neg P_1(g(x)) \vee \neg P_2(x) \vee P_3(g(y)) \vee P_4(y) \quad P_1(g(g(c))) \vee \neg P_5(z) \vee P_6(g(z))}{\neg P_2(g(c)) \vee P_3(g(y)) \vee P_4(y) \vee \neg P_5(z)) \vee P_6(g(z))}$$

The inference is based on the the mgu $\sigma = \{x \mapsto g(c)\}$. Rewriting these clauses to use $\rightarrow$, $\vee$ and $\wedge$ and applying σ eagerly to the premises we obtain the equivalent inference:

$$\frac{P_1(g(g(c))) \wedge P_2(g(c)) \rightarrow P_3(g(y)) \vee P_4(y) \quad P_5(z) \rightarrow P_1(g(g(c))) \vee P_6(g(z))}{P_2(g(c)) \wedge P_5(z)) \rightarrow P_3(g(y)) \vee P_4(y) \vee \vee P_6(g(z))}$$

To avoid accidental variable capture, the premises of the resolution rule are typically assumed to share no variables. Of course, it is safe to rename variables whenever required.

The calculus $\mathsf{R}_S^{\succ}$ is refutationally complete: a saturated set of clauses (i.e., closed by the inference rules of Figure 2.1) is unsatisfiable if and only if it contains the empty clause. The goal of a resolution based theorem

[1] An ordering $\succ$ on a set X is extended to an ordering $\succ_{mul}$ on finite multisets of X as follows: $\Sigma_1 \succ_{mul} \Sigma_2$ if i) $\Sigma_1 \neq \Sigma_2$ and ii) whenever $\Sigma_2(x) > \Sigma_1(x)$ then $\Sigma_1(y) > \Sigma_2(y)$, for some y such that $y \succ x$. Here $\Sigma(x)$ stands for the number of times x occurs in a multiset Σ. The multiset extension of an ordering is a standard notion; see, e.g., [Baader and Nipkow, 1998], for more information.

$$\text{Paramodulation}\quad \frac{C \vee s = t \quad D}{(C \vee D(t)_p)\sigma} \qquad \text{Reflexivity}\quad \frac{C \vee s \neq s}{C}$$

i. $\sigma = mgu(s, D|_p)$, where $D|_p$ is the subterm of D at position p.

ii. $D|_p$ is not a variable ([Brand, 1975]).

iii. $D(t)_p$ is the result of replacing in D the subterm at position p by t.

Figure 2.2: The (unordered) paramodulation rules.

prover is to derive a contradiction (in the form of an empty clause) from an unsatisfiable set of input clauses and the derivation process amounts to a saturation of the input set. In this respect, resolution is a saturation-based, refutational method.

The specific ordering and selection function used determine the inferences that can occur. Certain heuristics (e.g., heuristics that ensure termination on decidable fragments) can be defined by picking an admissible ordering together with a particular selection function. The completeness result for $\mathsf{R}_S^\succ$ guarantee the completeness of these heuristics.

One could combine $\mathsf{R}_S^\succ$ with the classical equality axioms of first-order logic (reflexivity, symmetry, congruence, etc.) to reason in a language that contains equality. However, this is known to be impractical. Robinson and Wos [1969] introduced the *paramodulation rule* and showed that in combination with $\mathsf{R}_S^\succ$ and some additional reflexivity axioms, it is refutationally complete for first-order logic with equality. It was later shown by Brand [1975] that many reflexivity axioms were not required for completeness, as well as paramodulation *into variables* (cf. condition ii. in Figure 2.2).

The rules of Figure 2.2 combined with resolution and positive factoring are enough for completeness. In fact, it is interesting to observe that in this setting the resolution rule is no longer needed since everything can be done equationally: take any first-order atom l and use $l = \mathit{true}$ and $l \neq \mathit{true}$, for a suitable constant true, instead of l and $\neg l$.

However, the paramodulation rule is difficult to control unless additional refinements are considered. An important tool in restricting the number of inferences is the use of *term orderings*. For example, *superposition* is the restricted version of paramodulation in which inferences only consider the biggest term (wrt an ordering $\succ$) of an equation. Bachmair and Ganzinger [1994] proved the completeness of an inference system for full first-order clauses with equality, based on *strict superposition*: paramodulation involving only maximal terms of maximal equations of clauses.

For further details on paramodulation based reasoning, the reader is referred to [Nieuwenhuis and Rubio, 2001].

$$\mathsf{Deletion}\quad \frac{N \cup M}{N}\ \dagger \qquad \mathsf{Deduction}\quad \frac{N}{N \cup M}\ \ddagger$$

Side conditions

† $M \subseteq \mathcal{R}(N)$.

‡ $M \subseteq \mathsf{C}(\Gamma(N))$ where $\mathsf{C}(\Gamma(N))$ are the consequents of $\Gamma(N)$.

Figure 2.3: Derivations for a calculus Γ with redundancy criterion $\mathcal{R}$.

A straightforward, naive implementation of a saturation-based calculus in which one exhaustively applies inferences to previously derived clauses will be hopelessly inefficient in all but the most trivial cases. In a clausal, saturation-based, refutational theorem-prover, each derived clause is a potential partial derivation of the empty clause and is increasing the search space. A partial derivation of the empty clause that is subsumed by another one is redundant and should be deleted to avoid useless computations. In most refutational provers, the deductive core accounts for a rather small part of the system, while most of its complexity derives from the implementation of redundancy elimination and simplification techniques.

Now, while redundancy elimination techniques are crucial from a practical point of view, it is not a priori clear to what extent they can be performed without compromising refutational completeness. Bachmair and Ganzinger [2001] address this issue by introducing a theoretical framework for saturation-based theorem proving. We give a very short overview of this framework next.

Theorem proving is presented as a series of *derivations* that transform a set of clauses[2] by additions and removals. The rules of derivation are shown in Figure 2.3. Observe that this is a very general (and abstract) presentation, where the calculus is represented by a map Γ from a set of clauses to the set of all inferences that may be drawn from those clauses and $\mathcal{R}$ is a *redundancy criterion* that maps a set of clauses N to a set of clauses *deemed redundant* by N. There are certain theoretical properties that a redundancy criterion must satisfy (e.g., "removing redundant clauses from an unsatisfiable set preserves unsatisfiability") that we won't list here.

A clause set is said to be *saturated up to redundancy* (with respect to Γ and $\mathcal{R}$) if all inferences in Γ with non-redundant premises are redundant in N, i.e., $\mathsf{C}(\Gamma(N \setminus \mathcal{R}(N))) \subseteq \mathcal{R}(N)$. It can be shown that saturation up to redundancy can be achieved by "fair" derivations, that is, intuitively, derivations where no non-redundant inference is delayed indefinitely.

[2]We will only consider here the case for *ground* clauses. For general clauses, an additional lifting has to be done.

Bachmair and Ganzinger [2001] also define what they call the *standard redundancy criterion* and give very general conditions under which a refutationally complete calculus Γ induces a system that is *derivationally complete*, i.e., every unsatisfiable set saturated up to redundancy contains the empty clause. In their words, the standard redundancy criterion "justifies most, if not all, of the common simplification and deletion techniques used in refutational theorem provers".

Standard redundancy applies to ordered calculi like $\mathsf{R}_S^{\succ}$. Intuitively, a clause C is said to be *redundant with respect to a set N of clauses* if there exist clauses $C_1, \ldots C_k$ in N such that $C_1, \ldots, C_k \models_{\mathrm{FO}} C$ and $C \succ C_i$ for all $1 \leq i \leq k$.

As an example, observe that standard redundancy already comprehends the most classical form of redundancy elimination: the *subsumption principle* [Robinson, 1965]. It is said that a clause C *subsumes* a clause D whenever $C\sigma \subseteq D$, for some substitution σ. The subsumption principle says that if a set N of clauses contains two distinct non-empty clauses C and D, and C subsumes D, then N is satisfiable iff $N \setminus D$ is too. For the ground case, if C subsumes D, then we have $C \subset D$ which implies $C \models_{\mathrm{FO}} D$ and, because $\succ$ is a multiset extension, $D \succ C$. Therefore, D is redundant with respect to N. The non-ground case is similar but requires a suitable lifting.

Now, what are the general conditions that guarantee that the standard redundancy criterion turns a refutationally complete system into a derivationally complete one? What Bachmair and Ganzinger [2001] show is that any calculus that possess what they call the *reduction property for counterexamples* is refutationally complete and induces a derivationally complete system in conjunction with the standard redundancy criterion. The upshot is that given a saturation-based refutational calculus, if one can establish its completeness by proving the reduction property for counterexamples, one gets an "adequacy for implementations" result for free. In a way, it can be regarded as an *scheme* of completeness proof that comes with an added value.

In Part III of this thesis we will prove completeness of various calculi following this scheme. Completeness proofs in general tend to be long and technical, and the ones we will present are no exception, but being familiar with the proof-scheme used makes them easier to follow. Therefore, we close this section with an overview of this method.

We will take a rather abstract view. First, clauses are just sets of ground formulas (each clause represents the disjunction of the formulas it contains). Second, a calculus is presented as a collection of inference rules; when a rule has more than one premise, we assume that one of them is tagged as the main premise, the others will be called side premises. Finally, we take as given a satisfaction relation $\models$ defined between models and formulas, clauses, etc.

A set of clauses N is called *saturated with respect to a set of rules* R if every clause obtained from N by the application of one of the rules in R is already in N. A set of clauses N is *inconsistent with respect to* R whenever the saturation of N with respect to R contains the empty clause, otherwise it is *consistent*. A saturation-based calculus R is *refutationally complete* (or *complete* for short) if every unsatisfiable set of clauses is inconsistent with respect to R.

A completeness proof can be reduced to showing that every saturated consistent clause set is satisfiable, i.e., has a model. One of the main ingredients of this proof-scheme is a procedure that builds a model (termed *candidate model*) from any consistent (but not necessarily saturated nor satisfiable) set of clauses N. Let us call I_N the model obtained from a consistent set N using such a procedure. What is ultimately shown is that if $I_N \not\models N$, then N is not saturated.

To prove this, the proof-scheme relies on a well-founded total ordering on clauses $\succ_c$ (e.g., the admissible ordering mentioned above). These two conditions guarantee that whenever N is consistent and $I_N \not\models N$, then there exists a minimum (with respect to $\succ_c$) clause $C \in N$ such that $I_N \not\models C$; we call C the *minimum counterexample of* I_N. The *reduction property for counterexamples* (with respect to $\succ$ and the candidate model building procedure) holds if for every N and every minimum counterexample C of I_N, there exists an inference from N with main premise C and a conclusion D such that $C \succ D$ and D is also a counterexample of I_N. Observe that this trivially implies that N is not saturated and, therefore, this property implies refutational completeness.

The proof of this property is typically split in two lemmas which trivially imply it:

Lemma 1 (Main premise reduction lemma)**.** On every inference rule, every consequent is smaller than its main premise.

Lemma 2 (Counterexample lemma)**.** Let N be consistent, and let $C \in N$ be a clause such that $I_N \not\models C$; then there exists some valid inference with C as main premise and the side premises, if any, in N, such that for some consequent D, $I_N \not\models D$.

We haven't said much about I_N yet. In a classical first-order setting, *Herbrand models* are used [Herbrand, 1930]. These are models whose domain is the set of (ground) syntactic terms of the language and that may be represented succinctly by a set of positive ground literals. Therefore, building a model from N in the classical setting amounts to generating a set of positive ground literals occurring in N.

The model building procedure is relatively simple. Every clause $C \in N$ contributes *at most* one formula to I_N. Let ε_C be what C contributes to the model I_N, then ε_C is either a singleton set containing a (positive) literal, or the empty set. When ε_C is not empty, C is said to be a *productive clause*. Now, it is important to observe that the content of ε_C depends on the contributions of the clauses which are $\succ_c$-smaller than C. Its precise definition has to be carefully tailored in order to prove the *Counterexample lemma*. The set of contributions of clauses $\succ_c$-smaller then C we just mentioned (i.e. $\bigcup_{C \succ D} \varepsilon_D$) plays an important role throughout the proof and is usually called I_C.

The general structure of the completeness proof is essentially simple (and very elegant), but carrying it out usually relies on the following two rather technical lemmas:

Lemma 3 (Downwards preservation lemma)**.** If $I_N \not\models C$, then $I_C \not\models C$.

Lemma 4 (Upwards preservation lemma)**.** Let D be the consequent of an inference whose main premise is C. If $I_N \not\models C$ and $I_C \not\models D$, then $I_N \not\models D$.

Let's see the role these two lemmas usually play. Given a clause C such that $C \in N$ and $I_N \not\models C$, using the *Downwards preservation lemma* we can conclude that it is also the case that $I_C \not\models C$. But since $I_C = \bigcup_{C \succ D} \varepsilon_D$, one needs to show that some productive clause D must have contributed a formula to I_C that "made" C not true (of course, the details of this vary with every proof). This fact should make D a suitable side premise for an inference from C such that for at least one consequent E, $I_C \not\models E$. Finally, we need the *Upwards preservation lemma* to conclude $I_N \not\models E$.

For Part III this is all what we need as we will always be working with ground clauses. For first-order logic, the result for ground clauses must be lifted to general clauses (containing variables). This can be done by using *liftable orders*, i.e., orders which are invariant under substitution of variables by terms [Bachmair and Ganzinger, 2001].

2.2 Direct resolution for modal and hybrid logics

In Part II of this thesis we will see that it is possible to translate modal and hybrid formulas to first-order logic and apply resolution theorem proving techniques like the ones sketched in the previous section. But one can consider instead resolution based calculi that work directly on the source logic. This has been studied in the modal case by various authors, e.g., [Fariñas del Cerro, 1982, Mints, 1990, Enjalbert and Fariñas del Cerro, 1989, Auffray et al., 1990, Fitting, 1990].

For the hybrid case, Areces et al. [2001b] present the direct resolution calculus DR, which is refutationally complete for $\mathcal{H}(@, \downarrow)$. Part III of this thesis will be devoted to study some refinements of DR. In particular, we will be able to show that it remains complete if one restricts to ordered inferences. In fact, we will see it possesses the reduction property for counterexamples and therefore is amenable to efficient implementations.

$$\text{RES}\quad \frac{C \vee @_i\neg p \quad D \vee @_i p}{C \vee D} \qquad\qquad \text{REF}\quad \frac{C \vee @_i\neg i}{C}$$

$$\text{SYM}\quad \frac{C \vee @_j i}{C \vee @_i j} \qquad\qquad \text{PAR}\quad \frac{C \vee \varphi(i) \quad D \vee @_i j}{C \vee D \vee \varphi(i/j)}$$

$$\wedge\quad \frac{C \vee @_i(\varphi_1 \wedge \varphi_2)}{C \vee @_i\varphi_1 \quad C \vee @_i\varphi_2} \qquad\qquad \vee\quad \frac{C \vee @_i(\varphi_1 \vee \varphi_2)}{C \vee @_i\varphi_1 \vee @_i\varphi_2}$$

$$[r]\quad \frac{C \vee @_i[r]\varphi \quad D \vee @_i\langle r\rangle j}{C \vee D \vee @_j\varphi} \qquad\qquad \langle r\rangle\quad \frac{C \vee @_i\langle r\rangle\varphi}{C \vee @_i\langle r\rangle j \quad C \vee @_j\varphi}\ \dagger$$

$$@\quad \frac{C \vee @_i@_j\varphi}{C \vee @_j\varphi} \qquad\qquad \downarrow\quad \frac{C \vee @_i{\downarrow}j.\varphi}{C \vee @_i\varphi(j/i)}$$

Side conditions

† $j \in \mathsf{Nom}$ is *fresh*.

Figure 2.4: The direct resolution calculus DR.

Like in the resolution calculus for first-order logic, the DR calculus works on sets of *clauses*. A clause, in this context, is a (finite) disjunction of *arbitrary* $\mathcal{H}(@, \downarrow)$ @-formulas. That is, unlike the first-order case, formulas in a clause need not be literals. For simplicity, we will assume clauses contain no repeated disjuncts and will identify every formula $@_i\varphi$ with the (singleton) clause $@_i\varphi \vee \bot$ and $\bot$ with the empty clause.

Of course, restricting to @-formulas is not an expressivity limitation in terms of satisfiability: a formula φ is satisfiable iff for some arbitrary nominal i not occurring in φ, $@_i\varphi$ is satisfiable.

The rules of DR are shown in Figure 2.4. Here, i and j are arbitrary nominals, p stands for a proposition symbol, $\varphi, \varphi_1, \varphi_2$ are arbitrary $\mathcal{H}(@, \downarrow)$-formulas and C and D are arbitrary clauses. In the antecedent of the PAR rule, $\varphi(i)$ indicates that the nominal i appears in φ. Observe that this presentation assumes formulas are in negation normal form.

We can group the rules in Figure 2.4 according to their role. Rules $\wedge$, $\vee$, @ and $\downarrow$ handle formula decomposition. Rule $\langle r\rangle$ is a form of skolemization, assigning a new name (through a new nominal) to an element of the model which was existentially quantified. Rule RES is a ground version of the resolution rule for first-order logic, while rule $[r]$ encodes a non-trivial unification plus a resolution step. Finally, rules SYM, REF and PAR handle equalities; observe that the latter is essentially the paramodulation rule of first-order logic.

Rule $\langle r \rangle$ is clearly satisfaction-preserving (i.e., if a model satisfies the antecedent, then there is a model that satisfies the consequent); the rest of the rules preserve satisfaction even in the same model (i.e., if a model satisfies the antecedent, then the same model satisfies the consequent), in the case of rule $\downarrow$ under the assumption that i does not occur bound in φ.

Given an $\mathcal{H}(@, \downarrow)$-formula φ, we define $\mathit{ClSet}(\varphi) = \{@_i\varphi\}$, for i an arbitrary nominal not occurring in φ. $\mathit{ClSet}^*(\varphi)$, the saturated set of clauses for φ, is then defined as the smallest set that includes $\mathit{ClSet}(\varphi)$ and is closed under the rules of DR. Areces et al. [2001b] show that the construction of $\mathit{ClSet}^*(\varphi)$ is a correct and complete (alas, hopelessly inefficient) algorithm for the satisfiability problem of $\mathcal{H}(@, \downarrow)$: φ is unsatisfiable if and only if the empty clause $\bot$ is an element of $\mathit{ClSet}^*(\varphi)$.

The size of $\mathit{ClSet}^*(\varphi)$ cannot be bounded in terms of the subformulas of φ because each application of rule $\langle r \rangle$ introduces a new nominal. Thus, the construction of $\mathit{ClSet}^*(\varphi)$ is not necessarily a decision procedure for the decidable logic $\mathcal{H}(@)$. In Part III we will see that, in fact, it is possible to obtain infinite derivations (even on a restricted version of the calculus).

We haven't said much, so far, on how we will use all these techniques in the context of this thesis. That will be the subject of the following chapter.

Chapter 3

What this thesis is about

One of the unifying themes of this thesis is the satisfiability problem for the hybrid logic $\mathcal{H}(@, \downarrow)$ and some of its sublogics, most notably, $\mathcal{H}(@)$. It seems therefore appropriate to discuss now some of the previous work on satisfiability (or validity) for hybrid logics.

Seligman [2001] developed a sound, complete and cut-free sequent calculus for $\mathcal{H}(@, \downarrow)$ by a series of transformations from a sequent calculus for first-order logic. This technique is quite general and can be applied to derive sequent calculi for a wide range of modal and hybrid logics. Although the resulting calculus does not posses the *Subformula Property* (a proof may contain formulas not occurring as subformulas in the end sequent), a weaker form of this property can be exploited in automating proof-searches. Earlier proof-theoretical results for what we would now recognize as hybrid languages, also based on sequent calculi, can be found in [Seligman, 1991, 1997]

Natural deduction systems for hybrid logics were investigated by Braüner [2004a,b]. In the first paper, he introduces a sound, complete and normalizing calculus for the very expressive hybrid logic $\mathcal{H}(@, \downarrow, \forall)$ (i.e., the extension of $\mathcal{H}(@, \downarrow)$ with the classical $\forall$ binder of first-order logic). In the second one, this system is compared to one due to Seligman [1997]. Although several results could be transferred from one system to the other, normalization of the latter remains an open problem.

Tableaux calculi pervade in modal logics so it is not surprising to find several tableaux systems that have been developed for hybrid logics. Tzakova [1999] presented Fitting-style prefix tableaux, for the expressive $\mathcal{H}(@, \downarrow, \forall)$. The paper also discusses a tableaux-based decision procedure for $\mathcal{H}(@)$. Blackburn [2000] exploits nominals and the satisfaction operator @ to develop labeled tableaux that require no additional meta-logical support (in the form of *labels*, *prefixes*, etc). These kind of tableaux are now known as *internalized tableaux*. This form of tableaux were extended to other systems (including the hybrid equivalents of *quantified modal logics*) by Blackburn and Marx [2002]. Bolander and Brauner [2006] presented terminating search strategies for some of the prefixed and internalized tableaux systems we just mentioned. These strategies are based on, so-called, *loop-checks* –a well-known technique from the *description logics* community– and apply to the logic $\mathcal{H}(@, \mathsf{A})$ (i.e., the logic that contains nominals, @ and the *universal modality* A that satisfies $\mathcal{M}, w \models \mathsf{A}\varphi$ iff $\mathcal{M}, v \models \varphi$ for all $v \in |\mathcal{M}|$; we will revisit this operator in Part IV). Later, Bolander and Blackburn [2007, 2009] extended these results by developing terminating tableaux strategies for more expressive hybrid logics (e.g., converse modalities, transitive frames, etc.).

Kaminski and Smolka [2009] show that loop-checking can be replaced in, many cases, by a saturation condition that is local (i.e., involves only information about one node of the tableaux). They refer to this approach as *pattern-based blocking*. It is shown that even the hybrid logic with the difference modality (which subsumes the universal modality) can be decided using pattern-based blocking, although for the converse modality, a form of loop-checking is needed.

It is well-known that there exists a close link between hybrid logics and *description logics* that include the “one-of” concept constructor, which can be seen, from a modal viewpoint, as a disjunction of nominals. Although nominals were known to fill an important expressivity gap in description logics, it was not clear how to incorporate them to the mix in an effective way. Nominals interact “badly” with inverse roles and number restrictions, leading to the loss of the finite model property and even very weak forms of the tree model property. Only recently Horrocks and Sattler [2007] proposed a tableaux-based, goal directed, decision procedure for the very expressive description logic $\mathcal{SHOIQ}$. Kazakov and Motik [2008], on the other hand, exhibit a decision procedure for $\mathcal{SHOIQ}$ using a translation to first-order logic and basic superposition extended with some non-standard simplification rules.

In this thesis, we will approach the satisfiability problem for $\mathcal{H}(@, \downarrow)$ using first-order resolution and variations of the direct resolution calculus described in Chapter 2. In a way, we can say that *resolution* is the second unifying theme of this thesis. It must be observed, though, that both techniques (i.e., first-order resolution via formula translations vs. direct resolution) are quite different in nature and differ in their motivations. In the following sections we describe in more detail our motivations and outline our results.

3.1 Translation-based methods

An efficient automated theorem prover for an expressive logic is a complex piece of software. Its performance crucially depends on the choice of datastructures and the nature and quality of the implemented heuristics and optimizations. In order to spot such opportunities for optimizations, one needs a profound knowledge of the underlying logic.

And efficiency is only half of the story. Innocent looking heuristics and optimizations may interact badly with each other, compromising the overall correctness. A combination of theoretical analysis with thorough, continuous testing is needed to avoid these pitfalls.

Before embarking in the adventurous task of writing an automated theorem prover of his own, a logic engineer might want to consider potentially more cost-effective alternatives, and this is the motivation for the first part of this thesis.

Propositional modal logics are usually less expressive than first-order logic. This means that we can find a (computable) translation of modal formulas to first-order formulas such that φ is satisfiable if and only if its translation φ^* is first-order satisfiable. Such translation is called *satisfiability preserving*.

On the other hand, state-of-the-art resolution-based theorem provers for first-order logic are –together with *SAT solvers* for propositional logic– the most advanced automated reasoners available today.

By composing a computable satisfiability preserving translation with a first-order logic prover, we get a sound and complete theorem prover for our logic for free. From a practical point of view, we shall only be interested in polynomial (preferably linear) time translations[1].

Now, since many modal logics are decidable and first-order logic is undecidable (in general), we are, a priori, reducing a decidable problem to an undecidable one. Theoretically, this is not necessarily the case; for example, the basic modal logic can be embedded in the two-variable fragment of first-order logic, which is known to be decidable [Mortimer, 1975]. However, the decidability of the fragment does not imply that every complete theorem prover for first-order logic will be able to decide it. In practice, we may end up obtaining a semi-decision procedure instead of a decision procedure.

Part II of this thesis will review several known families of translations of modal logics to first-order logic and try to adapt each to the hybrid setting.

We begin in Chapter 4 visiting translations based on the idea of encoding, as first-order logic formulas, the conditions that make a formula a *theorem* in a Hilbert-style system for modal logics. Because of strong completeness of these systems, this kind of encodings can be used to decide, using a first-order prover, if a modal formula is valid with respect to any class of frames that is modally definable; and this includes non-elementary classes of frames. Regrettably, it is known that in practice this technique gives very

[1]Interestingly, Sebastiani and Vescovi [2009] report on some experiments using state-of-the-art propositional SAT solvers in combination with a translation of $\mathcal{ML}$-formulas to propositional ones which can be exponentially larger and conclude that this can also be a feasible approach.

poor results already at the propositional level [McCune and Wos, 1992]. We will only briefly develop the hybrid case, but will pay attention to one point: the first-order encoding of hybrid theoremhood requires us to use the equality symbol. In a way, this is not surprising since it was claimed in Chapter 1 that hybrid logics add identity (and, therefore, notions of equality) to classical modal logics. However, while equality is easily seen in the semantics, it was not that evident, we believe, where it lied when looked from a Hilbert-style proof-theoretic point of view. The translation developed in this chapter makes clear this fact.

Chapter 5 is devoted to first-order encoding of modal and hybrid semantics. This is usually known as the *standard translation* of modal logics to first-order logic. It is known that even for the basic modal case, the standard translation behaves rather poorly in combination with a resolution-based theorem prover. The *layered translation* proposed by Areces et al. [2000] overcomes some of these limitations by exploiting the tree-model property to obtain a translation that "helps" a resolution-based prover to avoid many unnecessary inferences. This is achieved by introducing additional relation symbols that block unification of certain clauses. Hybrid logics don't posses the tree-model property; we will nonetheless develop a layered translation for $\mathcal{H}(@, \downarrow)$ that follows the spirit of the one for the basic modal logic.

Finally, in Chapter 6 we turn to the so-called *functional translations*. These can be seen as the first-order encoding of an alternative semantics for modal logics based on *functional models*, that is, models where sets of functions are used instead of relations. These translations are known to behave well in combination with resolution-based theorem provers, in particular the *optimized functional translation* which requires the use of skolem constants only. We will see in this chapter that hybrid logics fit in quite naturally in this framework.

The functional translations are typically presented using a sorted logic. Arguably, they simplify their presentation but they may also introduce a slight increase in complexity. We finish Chapter 6 by discussing the need for sorts when using the (optimized) functional translation. We will see that for reasoning with respect to any class of models that is modally definable, sorts are not required (i.e., they can be simply removed from translated formulas). However, we will show that this is not the case in general and, in particular, we will exhibit a class of models that is definable with a hybrid formula where removing sorts in this way is unsound.

3.2 Direct resolution

In Part III of this thesis we turn to the direct resolution calculus DR of Areces et al. [2001b] that we already introduced in Chapter 2. This calculus is described by its authors as a blend of the labeling method used in tableaux with resolution and paramodulation, that is, a combination of two techniques, tableaux and resolution, that proved to be efficient in dealing with modal logics and first-order logic with equality, respectively. Since hybrid logics can be described as "modal logics with equality", DR seems like an interesting basis for automated reasoning in this context. However, since DR incorporates no mechanism to regulate the generation of clauses it is, in this form, of little practical interest.

In first-order logic, this regulation is achieved by relying on orderings on formulas, as is illustrated by the ordered resolution calculus $\mathsf{R}_S^{\succ}$ described in Chapter 2. Intuitively, the general idea is to show that one can *choose* a literal from each ground clause such that rules are to be applied to a clause only to eliminate its chosen literal. In $\mathsf{R}_S^{\succ}$, the chosen ground literal is the maximum one according to $\succ$ unless it is overriden by the *selection function* S. As long as $\succ$ satisfies certain "admissibility" conditions, $\mathsf{R}_S^{\succ}$ is refutationally complete.

Inspired by the first-order case, our contribution in Chapter 7 is to turn DR into a calculus of ordered resolution with selection functions. The resulting calculus, $\mathsf{DR}_S^{\succ}$, will be parameterized by an ordering on formulas $\succ$ and a suitable selection function S and we will give sufficient admissibility conditions on $\succ$ to guarantee its refutational completeness. We will, in fact, prove the stronger result that $\mathsf{DR}_S^{\succ}$ possesses the *reduction property for counterexamples* and is therefore compatible with the standard redundancy criterion (cf. Chapter 2). We will be, therefore, establishing the theoretical basis for the development of realistic provers for hybrid logics with direct resolution as deductive core.

The proof of the reduction property for counterexamples we give follows the scheme outlined in Chapter 2. Observe that in order to pursue it, we have to provide an adequate notion of *Herbrand model* for hybrid logics, which is also done in Chapter 7.

In Chapter 8 we will introduce two successive refinements of $\mathsf{DR}_S^{\succ}$. First, we obtain $\mathsf{DRL}_S^{\succ}$ which, conceptually, differs from $\mathsf{DR}_S^{\succ}$ in that the paramodulation rule only needs to replace the nominal occurring in the *label* of a formula. Given that paramodulation is a computationally expensive rule, restricting its application to only labels in formulas in the clause set should result in improved performance.

Establishing the refutational completeness of $\mathsf{DRL}_S^{\succ}$ proved to be a challenging task. As will be discussed in Chapter 8, in this case it is simply not possible to follow the proof scheme outlined in Chapter 2, unless one replaces the usage of the satisfiability relation $\models$ during the proof in favor of a *stronger* relation we termed $\models\!\!\!\approx$.

The second calculus introduced in Chapter 8 is called $\mathsf{DRL}\epsilon_S^{\succ}$ and it is shown to be complete for $\mathcal{H}(@,\downarrow)$ and terminating for $\mathcal{H}(@)$ (that is, the smallest set that contains an $\mathcal{H}(@)$-formula and is saturated by the rules of $\mathsf{DRL}\epsilon_S^{\succ}$ is always finite). This means that any implementation of $\mathsf{DRL}\epsilon_S^{\succ}$

(even one without any redundancy elimination techniques) will constitute a decision procedure for the satisfiability problem of $\mathcal{H}(@)$.

The main difference between $\mathsf{DRL}\epsilon^{\succ}_{S}$ and $\mathsf{DRL}^{\succ}_{S}$ lies in the way on-the-fly skolemization is performed. While the latter simply introduces a fresh nominal every time (similar to what is done in DR), the former relies on an additional sort of atomic symbols, called ϵ-terms. From a first-order logic point of view, these are an alternative to skolem functions, that can be used in tableaux systems [Giese and Ahrendt, 1999]. We will see that, in our case, they provide us with a natural notion of *derivation history* that we can use to avoid the generation of an unbounded number of clauses.

Finally, in Chapter 9 we will discuss some lessons learnt from the implementation of a prototypic automated theorem prover based on $\mathsf{DRL}\epsilon^{\succ}_{S}$.

3.3 Coinduction, extractability, normal forms

The results in the last part of this thesis were motivated by the search for normal forms that simplify the task of automated reasoners for hybrid logics. Be that as it may, it must be said that this quest took us in some unexpected directions.

In automated deduction, transformations to normal form often serve to simplify the formulation of inference calculi. Throughout Parts II and III, for example, we will often assume formulas to be in negation normal form to simplify definitions and proofs. Another example is the clause normal form which is used in classical resolution (cf. Chapter 2) but also in the Davis-Putnam algorithm, clausal first-order tableaux, etc.

Clausal normal forms are ubiquitous in automated reasoning, and this means formulas often have to be transformed into this form beforehand. There are many ways in which this transformation may be performed; in practice, it makes a lot of difference the transformation used. For starters, if we are willing to obtain a formula equivalent to the initial one, then the clause normal form transformation may result in a transformed formula that is exponentially larger. Typically one only needs an *equisatisfiable* formula in clausal form, and there are many such transformations with only a linear overhead. Nonnengart and Weidenbach [2001] review clause normal form transformations for first-order logic that aim to: i) reduce the final size, and ii) simplify the resulting formula by removing trivial tautologies, redundancies, etc.

Our investigations began when considering formulas like this one:

$$\varphi \wedge [r]@_i\psi \tag{3.1}$$

We observed that, because of the way labeled calculi work, the subformula $@_i\psi$ is in certain way "trapped" inside the box $[r]$ and this may lead to its repeated processing. To help clarify this idea, consider Figure 3.1, where

C_1:	$@_j\,((p_1 \vee \langle r\rangle k_1) \wedge (p_2 \vee \langle r\rangle k_2) \wedge [r]@_i\psi)$	
C_2:	$@_j(p_1 \vee \langle r\rangle k_1)$	(by $\wedge$ on C_1)
C_3:	$@_j(p_2 \vee \langle r\rangle k_2)$	(ditto)
C_4:	$@_j[r]@_i\psi$	(ditto)
C_5:	$@_j p_1 \vee @_j\langle r\rangle k_1$	(by $\vee$ on C_2)
C_6:	$@_j p_2 \vee @_j\langle r\rangle k_2$	(by $\vee$ on C_3)
C_7:	$@_j p_1 \vee @_{k_1}@_i\psi$	(by $[r]$ on C_4 and C_5)
C_8:	$@_j p_1 \vee @_i\psi$	(by $\langle r\rangle$ on C_7)
C_9:	$@_j p_2 \vee @_{k_2}@_i\psi$	(by $[r]$ on C_4 and C_6)
C_{10}:	$@_j p_2 \vee @_i\psi$	(by $\langle r\rangle$ on C_8)

Figure 3.1: From $[r]@_i\psi$ (C_1) to two copies of $@_i\psi$ (C_8 and C_{10}).

a (partial) derivation using the rules of the calculus DR (cf. Chapter 2) is shown. Every time rule $[r]$ is applied, a new "copy" of $@_i\psi$ is obtained. The end result, in this example, is that clauses C_8 and C_{10} both contain $@_i\psi$; hence, the rules of the calculus needed to, say, reduce $@_i\psi$ will have to be applied twice. Clearly, we can modify this example to obtain as many copies of $@_i\psi$ as we wish. A similar example that illustrates this problem but in a tableau calculus would be:

$$@_j(((\langle r\rangle k_1 \vee \langle r\rangle k_2 \vee \cdots \vee \langle r\rangle k_n) \wedge [r]@_i\psi) \tag{3.2}$$

In this case, we will be forced to explore at least n branches, and on each branch $@_i\psi$ will have to be processed from scratch.

But $@_i\psi$ is a *global* formula, in the sense that its truth value does not depend on where in the model it is evaluated. If we apply whatever rules are needed on $@_i\psi$ more than once we are, in a way, wasting effort.

Now, observe that "globality" also means that (3.1) is logically equivalent to the following:

$$\varphi \wedge ([r]\bot \vee @_i\psi) \tag{3.3}$$

Since in (3.3) $@_i\psi$ does not appear in the scope of $[r]$, it is easy to ensure that $@_i\psi$ is effectively processed only once[2]. We will typically say that $@_i\psi$ was *extracted* from within the scope of $[r]$ and, therefore, call a formula in "extracted form" whenever no @-subformula is extractable.

We will define in Chapter 11 transformations to extracted form, both into equivalent and equisatisfiable formulas (one can assure there is no exponential blow-up only for the equisatisfiable transformation). But we will go beyond $\mathcal{H}(@, \downarrow)$. In a way, one question we will try to answer throughout

[2]In the case of direct resolution, we need to use selection functions (introduced in Part III) or a similar technique.

Chapter 11 is "what is so special about $@_i$ that allows us to *extract* it from other modalities?"

It turns out that even properly characterizing the concept of "extractability" is far from trivial. As we will see, one rather needs to consider "extractability" as a directed relation among the set of modalities in the language. We end Chapter 11 discussing complexity issues related to extracted forms.

A hybrid logic such as $\mathcal{H}(@, \downarrow)$ is usually called an *extension* of the basic modal logic $\mathcal{ML}$. We say it is an "extension" because additional semantic clauses are needed to give meaning to the additional modal operators $@_i$ and $\downarrow i$ (cf. Definition 1.7). We were interested in transferring our results to other extensions of the basic modal logic (e.g., memory logics, which can be seen as weaker forms of hybrid logics [Areces et al., 2008]), but since different extensions have different semantics (i.e., different semantic clauses), this was not possible without redoing the proofs in each case.

We found we can overcome this apparent limitation by expressing modal semantics in terms of a novel type of models that are coinductively defined. Intuitively, these models can be seen as a form of coalgebraic abstract datatype whose observable behaviour determines the semantics of the modal operators of the logic (Jacobs and Rutten [1997] wrote a nice introduction to coinduction and its links to coalgebras).

The basic modal logic is complete with respect to the class of all the coinductive models; moreover we can define many extended modal logics such as hybrid logics and memory logics by restricting our attention to particular classes of coinductive models. This way, results that had to be proved separately for each different language (but whose proofs were known to be mere routine) now can be proved in a general way. We will see, for example, that we can have a unique definition of bisimulation for all these languages, and prove a single invariance-under-bisimulation theorem.

This setting is presented in Chapter 10. Being outside of the scope of this thesis, we will just scratch its surface and develop only those basic notions required in Chapter 11. A more systematic study is left as future work.

Part II

Translation-based methods

Chapter 4

Theoremhood encodings

There is a long standing tradition of defining Hilbert-style systems for modal logics. A *theorem* in this kind of systems is either one of the formulas in a recursive set of axioms or is derived from them by finitely applying certain inference rules. It is known that in the modal case one can encode these derivability conditions using first-order logic *without equality*. This means that given a modal formula φ one can recursively obtain a first-order formula $\varphi^\star$ such that $\varphi^\star$ is satisfiable if and only if there exists a proof for φ.

Since there exist Hilbert-style modal systems that are sound and complete with respect to classes of models that are not first-order definable, these kind of translations enable the use of first-order provers in cases where translations based in modal semantics (like the ones we are going to present in Chapter 5) are of no use.

In this chapter we will begin by briefly reviewing this type of encoding and we will then discuss how it can be extended to the hybrid case. We will observe that the latter requires that we move to first-order logic *with equality*. This in a sense is not unexpected, after all, hybrid logics are sometimes referred to as "modal logics with equality". However, while, semantically, it is clear where equality comes from, it will be interesting to pinpoint, of all the axioms and inference rules added in a hybrid system, which are responsible for incorporating equality to the mix.

4.1 The case of classical modal systems

Let us start by considering a Hilbert-style system for propositional logic. Figure 4.1 shows such a system, which shall be referred to as P. It consists of three axioms (KN1-KN3) and two inference rules: *modus ponens* (MP) and *universal substitution* (Subst). One can find similar systems in the literature; they usually differ only in the choice of axioms. This one is due to Rosser [1953]; we have picked it because it has few and quite simple axioms.

Axioms

(KN1)	$\vdash_P p \to (p \land p)$
(KN2)	$\vdash_P (p \land q) \to p$
(KN3)	$\vdash_P (p \to q) \to \neg(q \land r) \to \neg(p \land r)$

Inference rules

(MP)	If $\vdash_P \varphi \to \psi$ and $\vdash_P \varphi$ then $\vdash_P \psi$
(Subst)	If $\vdash_P \varphi$ then $\vdash_P \varphi\sigma$ (σ substitutes propositions by formulas)

Figure 4.1: System P for the propositional calculus.

The set of *theorems of P* is the smallest set of propositional formulas that contains axioms KN1-KN3 and is closed by the rules MP and Subst. We write $\vdash_P \varphi$ whenever φ is a theorem of P.

Consider now the set of ground first-order terms built using constants $c_p, c_q, c_r, \ldots$ (that is, one constant per propositional variable) and function symbols $f_\neg$, $f_\land$ and $f_\to$, where the first one is unary and the rest are binary. There is a trivial bijection between first-order terms in this vocabulary and propositional formulas:

$$\begin{aligned} p^* &= c_p \\ (\neg\varphi)^* &= f_\neg(\varphi^*) \\ (\varphi \land \psi)^* &= f_\land(\varphi^*, \psi^*) \\ (\varphi \to \psi)^* &= f_\to(\varphi^*, \psi^*) \end{aligned}$$

One can easily show that the set of theorems of P, codified in this language, is first-order definable. That is, one can give a first-order theory Γ_P for a one-place relation symbol Thm such that, the following holds, for all propositional formula φ:

$$\vdash_P \varphi \text{ iff } \Gamma_P \models_{\text{FO}} \text{Thm}(\varphi^*) \tag{4.1}$$

For example, it is simple to encode (MP) adding this formula to Γ_P:

$$\forall x \forall y.((\text{Thm}(f_\to(x, y)) \land \text{Thm}(x)) \to \text{Thm}(y)) \tag{4.2}$$

Observe that when interpreted over any Herbrand model (that is, one whose domain of interpretation is the set of all ground terms), the universally quantified variables x and y in (4.2) range over all possible propositional formulas. This gives the intuitive justification for the encoding of (MP).

Similarly, the axioms of P could, in principle, be accounted for in a naive, straightforward manner; e.g., for (KN1) one could simply add to Γ_P this formula:

$$\mathrm{Thm}((p \to (p \wedge p))^*) \tag{4.3}$$

This naive approach would complicate matters when trying to encode rule (Subst), that requires us to substitute occurrences of propositional symbols by formulas. If we were to pursue this encoding of axioms, we would be forced to define a first-order theory of substitution. But since we are coding propositional formulas as first-order terms, we can exploit first-order logic's proof-theoretical instantiation mechanism, instead. The idea is simply to codify the three axioms of P using first-order variables instead of a constants c_p, c_q, and c_r.

Definition 4.1 (Γ_P)**.** We define, Γ_P, the first-order theory of P-derivability as the set containing the following formulas:

$$\forall x.\mathrm{Thm}(f_\to(x, f_\wedge(x, x))) \tag{4.4}$$
$$\forall x.\mathrm{Thm}(f_\to(f_\wedge(x, x), x)) \tag{4.5}$$
$$\forall xyz.\mathrm{Thm}(f_\to(f_\to(x, y), f_\to(f_\neg(f_\wedge(y, z)), f_\neg(f_\wedge(x, z))))) \tag{4.6}$$
$$\forall xy.((\mathrm{Thm}(f_\to(x, y)) \wedge \mathrm{Thm}(x)) \to \mathrm{Thm}(y)) \tag{4.7}$$

Observe that, for example, (4.4) is simply the universal closure of the formula obtained by replacing every occurrence of constant c_p in (4.3) by variable x.

Theorem 4.1. *For all propositional formula φ, $\vdash_P \varphi$ iff $\Gamma_P \models_{\mathrm{FO}} \varphi^*$.*

Time to move to the basic modal case. Figure 4.2 presents system K_r, which characterizes the set of valid formulas of the basic multimodal logic. A quick inspection shows that it differs from system P (Figure 4.1) only in the addition of one axiom and one inference rule: (K_r) and (Nec$_r$), respectively.

It is fairly straightforward to extend the encoding for system P to the case of K_r. First, we need to add a new unary function symbol $f_{[r]}$ to the language and make $(\cdot)^*$ a bijection with modal formulas adding the following clause:

$$([r]\varphi)^* = f_{[r]}(\varphi^*) \tag{4.8}$$

Finally, we need a first-order theory Γ_{K_r} for K_r-derivability, which we obtain quite simply from Γ_P by adding suitable formulas to account for (K_r) and (Nec$_r$).

Definition 4.2 (Γ_{K_r})**.** We define, Γ_{K_r}, the first-order theory of derivability in K_r as the set that extends Γ_P with the following formulas:

$$\forall xy.\mathrm{Thm}(f_\to(f_{[r]}(f_\to(x, y)), f_\to(f_{[r]}(x), f_{[r]}(y)))) \tag{4.9}$$
$$\forall x.(\mathrm{Thm}(x) \to \mathrm{Thm}(f_{[r]}(x))) \tag{4.10}$$

Axioms

(KN1)	$\vdash_{\mathsf{K}_r} p \to (p \wedge p)$
(KN2)	$\vdash_{\mathsf{K}_r} (p \wedge q) \to p$
(KN3)	$\vdash_{\mathsf{K}_r} (p \to q) \to \neg(q \wedge r) \to \neg(p \wedge r)$
(K_r)	$\vdash_{\mathsf{K}_r} [r](p \to q) \to [r]p \to [r]q$

Inference rules

(Nec$_r$)	If $\vdash_{\mathsf{K}_r} \varphi$ then $\vdash_{\mathsf{K}_r} [r]\varphi$
(MP)	If $\vdash \varphi_{\mathsf{K}_r} \to \psi$ and $\vdash_{\mathsf{K}_r} \varphi$ then $\vdash_{\mathsf{K}_r} \psi$
(Subst)	If $\vdash_{\mathsf{K}_r} \varphi$ then $\vdash_{\mathsf{K}_r} \varphi\sigma$ (σ substitutes propositions by formulas)

Figure 4.2: System K_r for the basic multimodal logic.

Theorem 4.2. *For every formula φ of the basic modal logic, it follows that $\vdash_{\mathsf{K}_r} \varphi$ iff $\Gamma_{\mathsf{K}_r} \models_{\mathrm{FO}} \varphi^*$.*

We have already sketched the idea of the proof of this theorem; more details can be found, for example, in [Rabe et al., 2009].

Many classical modal logics can be obtained by adding new axioms to this system. For example, if we add formula $[r]p \to [r][r]p$ to K_r as a new axiom, we obtain a system that is sound and complete with respect to the class of models where r is a transitive relation; while adding $[r]([r]p \to p) \to [r]p$ instead leads to a system sound and complete with respect to the class of those models where r is a finite trasitive tree. Notice that these are, respectively, formulas 4 and *Löb* from Figure 1.5. Clearly, we can extend Γ_{K_r} accounting for these new axioms by adding, respectively, the following formulas, which encode 4 and *Löb*, respectively, in the expected way:

$$\forall x.\mathrm{Thm}(f_{\to}(f_{[r]}(x), f_{[r]}(f_{[r]}(x)))) \tag{4.11}$$

$$\forall x.\mathrm{Thm}(f_{\to}(f_{[r]}(f_{\to}(f_{[r]}(x), x)), f_{[r]}(x))) \tag{4.12}$$

Observe that in the case of the Löb formula, this means we can prove in first-order logic modal validity with respect to a non-elementary class.

The main advantage of this translation is that it is simple to extend in order to handle additional modal axioms. However, experiments have shown that already for the propositional fragment the search space for this kind of encoding is very large [McCune and Wos, 1992]. In a way, finding a model for a first-order formula obtained by encoding derivability in a Hilbert-style system amounts to finding a proof in such a system, which is (because these systems are not goal-oriented) inherently inefficient.

Nevertheless, Rabe et al. [2009] recently report on successful results using automated theorems provers to establish the subset relationship between hundreds of modal axiomatizations using this encoding. This suggests that there may be some place for applications based on this approach.

4.2 Hybrid systems

Several Hilbert-style axiomatizations for $\mathcal{H}(@)$ and $\mathcal{H}(@,\downarrow)$ are known. We will use the ones due to Blackburn and ten Cate [2006]. We begin by considering the one for $\mathcal{H}(@)$, displayed in Figure 4.3. How does this system compare with system K_r in Figure 4.2? It is clearly an extension of it, that adds six new axioms and four new inference rules.

Observe first that one of the additional inference rules is also a substitution rule, (Subst$_{ij}$). Unlike (Subst), which replaces propositions by formulas, this one uniformly replaces nominals by nominals. The translations we reviewed in the previous section avoid explicitly having to encode the substitution rule and, of course, we would like to achieve the same with (Subst$_{ij}$). However, in order to enforce that nominals are not replaced by arbitrary formulas, we will need to take some additional provisions.

Of the other three inference rules, only (Name) and (BG) are worth mentioning. Because of their side-conditions, they are considered non-othodox rules[1]. Blackburn and ten Cate [2006] observe that this kind of rules are somewhat displeasing from an algebraic point of view but they show that, in the case of $\mathcal{H}(@)$ they cannot be replaced by a finite number of orthodox rules without losing completeness for all pure extensions.

In the side-conditions of both rules we can find an expression of the form $i \notin \mathit{Free}(\varphi)$, which is just a formal way of expressing that "i does not occur in φ". As we will later see, these conditions are the reason why we will have to introduce equality to the language when developing the first-order encoding.

Our main plan is to follow the same type of encoding used for K_r in the previous section. One of the problems we have to face now, is that of rule (Subst$_{ij}$). That is, we want to rely on first-order logic's instantiation mechanism, as we did with (Subst), but instantiation in this case must be of a different kind. We will see this is easy to achieve once we move to a

[1]Following Blackburn and ten Cate [2006], we consider a modal rule to be orthodox (in the presence of the modus-ponens rule) when it is of the form

$$\frac{\vdash \psi(\alpha_1,\ldots,\alpha_n)}{\vdash \varphi(\alpha_1,\ldots,\alpha_n)}$$

where $\alpha_1,\ldots,\alpha_n$ range over arbitrary formulas and are implicitly universally quantified. Of course, uniform substitution rules are excluded from this definition.

Axioms

(KN1)	$\vdash p \to (p \wedge p)$
(KN2)	$\vdash (p \wedge q) \to p$
(KN3)	$\vdash (p \to q) \to \neg(q \wedge r) \to \neg(p \wedge r)$
(K_r)	$\vdash [r](p \to q) \to [r]p \to [r]q$
($\mathsf{K}_@$)	$\vdash @_a(p \to q) \to @_a p \to @_a q$
(SD$_@$)	$\vdash @_a p \leftrightarrow \neg @_a \neg p$
(Ref$_@$)	$\vdash @_a a$
(Agree)	$\vdash @_a @_b p \leftrightarrow @_b p$
(Intro)	$\vdash a \to (p \leftrightarrow @_a p)$
(Back$_\mathrm{r}$)	$\vdash \langle r \rangle @_a p \to @_a p$

Inference rules

(MP)	If $\vdash \varphi \to \psi$ and $\vdash \varphi$ then $\vdash \psi$	
(Nec$_r$)	If $\vdash \varphi$ then $\vdash [r]\varphi$	
(Nec$_@$)	If $\vdash \varphi$ then $\vdash @_a\varphi$	
(Subst)	If $\vdash \varphi$ then $\vdash \varphi\sigma$	
(Subst$_{ij}$)	If $\vdash \varphi$ then $\vdash \varphi(i/j)$	
(Name)	If $\vdash @_i\varphi$ then $\vdash \varphi$	$(i \notin \mathit{Free}(\varphi))$
(BG$_r$)	If $\vdash @_i\langle r \rangle j$ then $\vdash @_j[r]\varphi$	$(j \neq i$ and $j \notin \mathit{Free}(\varphi))$

Notes:

- $\langle r \rangle \varphi$ is short for $\neg[r]\neg\varphi$.
- $\varphi \leftrightarrow \psi$ is short for $(\varphi \to \psi) \wedge (\psi \to \varphi)$.
- a and b are fixed nominals.

Figure 4.3: Hilbert-style system for the hybrid logic $\mathcal{H}(@)$.

many-sorted logic. Since it is simple to embed a sorted logic in classical (unsorted) logic, we won't lose generality.

In many-sorted logic, a sort symbol can be viewed as a unary predicate and denotes a subset of the domain of interpretation. Quantification can restrict variables to a sort; for example $\forall x{:}\alpha.\varphi$ and $\exists x{:}\alpha.\varphi$, with α a sort symbol, are interpreted as $\forall x.(\alpha(x) \to \varphi)$ and $\exists x.(\alpha(x) \wedge \varphi)$ respectively. Similarly, a function sort declaration like, $f : \alpha_1 \times \alpha_2 \to \alpha_3$ is interpreted as $\forall xy.(\alpha_1(x) \wedge \alpha_2(y) \to \alpha_3(f(x, y)))$ while a predicate declaration $P : \alpha_1 \times \alpha_2$ is interpreted as $\forall xy.(P(x, y) \to (\alpha_1(x) \wedge \alpha_2(y)))$. Finally, we allow for subsort declarations of the form $\alpha_1 \sqsubseteq \alpha_2$, which corresponds to the axiom $\forall x.(\alpha_1(x) \to \alpha_2(x))$. Refer to [Walther, 1987] for a thorough presentation of many-sorted logic.

Our approach will be clear once we define the sorted language we will work with.

Definition 4.3. We will work in a language with two sorts η and ρ, that satisfy $\eta \sqsubseteq \rho$. Terms will be formed with constants $c_p, c_q, \ldots$ of sort ρ; and $c_i, c_j, \ldots$ of sort η; together with the following function symbols:

$$\begin{array}{ll} f_\neg : \rho \to \rho & f_{[r]} : \rho \to \rho \\ f_\wedge : \rho \times \rho \to \rho & f_@ : \eta \times \rho \to \rho \\ f_\to : \rho \times \rho \to \rho & \end{array}$$

Intuitively, ρ is the sort of (arbitrary) formulas while η is the sort of nominals. Nominals are formulas as expressed by the constraint $\eta \sqsubseteq \rho$.

Function symbols $f_\neg$, $f_\wedge$, $f_\to$ and $f_{[r]}$ are simply those of the previous section adapted to this many-sorted setting. Observe that they combine arbitrary formulas into new formulas. The axioms and inference rules of K_r can be encoded almost exactly like was done before. For example, the inference rule (Nec$_r$) would be now encoded as:

$$\forall x{:}\rho.(\text{Thm}(x) \to \text{Thm}(f_{[r]}(x))) \tag{4.13}$$

Notice this formula differs from (4.10) only in the sort declaration for x. Observe also that because of the subsort declaration $\eta \sqsubseteq \rho$, the formula $\text{Thm}(c_i) \to \text{Thm}(f_{[r]}(c_i))$ for constant c_i of sort η is also a valid instance of (4.13).

Function symbol $f_@$ is new. It takes a nominal (sort η) and a formula (sort ρ). Intuitively, $f_@$ will be used to encode hybrid formulas of the form $@_i\varphi$.

Let us present a couple of examples in order to get a clearer picture of what we plan to do. Consider, for instance axiom (Ref$_@$). Since a is a nominal, inference rule (Subst$_{ij}$) may be applied and, therefore, we have that $\vdash @_i i$ holds for any i. We will encode this axiom as follows:

$$\forall x{:}\eta.\text{Thm}(f_@(x, x)) \tag{4.14}$$

$$
\begin{array}{rlrl}
p^* & \overset{def}{=} c_p & p^\star & \overset{def}{=} x_p \\
i^* & \overset{def}{=} c_i & i^\star & \overset{def}{=} x_i \\
(\neg\varphi)^* & \overset{def}{=} f_\neg(\varphi^*) & (\neg\varphi)^\star & \overset{def}{=} f_\neg(\varphi^\star) \\
(\varphi \land \psi)^* & \overset{def}{=} f_\land(\varphi^*, \psi^*) & (\varphi \land \psi)^\star & \overset{def}{=} f_\land(\varphi^\star, \psi^\star) \\
(\varphi \to \psi)^* & \overset{def}{=} f_\to(\varphi^*, \psi^*) & (\varphi \to \psi)^\star & \overset{def}{=} f_\to(\varphi^\star, \psi^\star) \\
([r]\varphi)^* & \overset{def}{=} f_{[r]}(\varphi^*) & ([r]\varphi)^\star & \overset{def}{=} f_{[r]}(\varphi^\star) \\
(@_i\varphi)^* & \overset{def}{=} f_@(i^*, \varphi^*) & (@_i\varphi)^\star & \overset{def}{=} f_@(i^\star, \varphi^\star)
\end{array}
$$

Figure 4.4: Translations $(\cdot)^*$ and $(\cdot)^\star$ take $\mathcal{H}(@)$-formulas to ground and non-ground first-order terms in the language of Definition 4.3.

Notice that x stands for things of sort η (nominals) and, therefore, cannot be instanced in an arbitrary formula (sort ρ). That is, while $\text{Thm}(f_@(c_i, c_i))$ is a valid instance of (4.14) for c_i of sort η, $\text{Thm}(f_@(c_p, c_p))$, for c_p of sort ρ, is not (in fact, it is not even a well-formed formula).

Before moving to the encoding of rules (Name) and (BG$_r$), it is worth formalizing what we discussed so far. Figure 4.4 shows two encodings of formulas of $\mathcal{H}(@)$ as ground and non-ground first-order terms, respectively. One of them corresponds to bijection $(\cdot)^*$ of the previous section, while the other one will be used to mechanically encode axioms, instead of doing it by hand, as until now. Notice that they only differ in the way atoms are handled: while $(\cdot)^*$ encodes them as constants, $(\cdot)^\star$ employs variables, instead. Now, let φ be any axiom of the system of Figure 4.3; then φ is encoded by taking the well-sorted universal closure of $\text{Thm}(\varphi^\star)$. Observe that, for example, (4.14) is the (well-sorted) universal closure of $\text{Thm}((@_a a)^\star)$.

Let us now take a look at the inference rule (Name). This rule looks similar to the converse of (Nec$_@$), except for its side-condition. This condition, $i \notin \mathit{Free}(\varphi)$, is not easy to enforce. We will have to introduce an additional predicate symbol $\mathit{Occurs}_\eta : \eta \times \rho$ and we will interpret $\mathit{Occurs}_\eta(y, x)$ as "the nominal y occurs free in formula x". Using this predicate, we can encode (Name) as follows:

$$\forall x{:}\rho \forall y{:}\eta.((\neg \mathit{Occurs}_\eta(y, x) \land \text{Thm}(f_@(y, x))) \to \text{Thm}(x)) \tag{4.15}$$

Using equality, it is possible to define a first-order theory that gives the desired meaning to Occurs_η. Figure 4.5 shows such a theory, which we call $\Sigma_{\mathit{Occurs}_\eta}$. Although $\Sigma_{\mathit{Occurs}_\eta}$ requires an infinite number of axioms (for there is a formula that must be instantiated for every constant c_p of sort ρ) this

$$\begin{aligned}
&\forall x{:}\eta\forall y{:}\eta.(\mathit{Occurs}_\eta(y,x) \leftrightarrow y = x)\\
&\forall y{:}\eta.(\neg\mathit{Occurs}_\eta(y,c_p)),\text{ for every constant } c_p \text{ of sort } \rho\\
&\forall x{:}\rho\forall y{:}\eta.(\mathit{Occurs}_\eta(y,f_\neg(x)) \leftrightarrow \mathit{Occurs}_\eta(y,x))\\
&\forall u{:}\rho\forall v\rho\forall y{:}\eta.(\mathit{Occurs}_\eta(y,f_\wedge(u,v)) \leftrightarrow (\mathit{Occurs}_\eta(y,u) \vee \mathit{Occurs}_\eta(y,v)))\\
&\forall u{:}\rho\forall v\rho\forall y{:}\eta.(\mathit{Occurs}_\eta(y,f_\rightarrow(u,v)) \leftrightarrow (\mathit{Occurs}_\eta(y,u) \vee \mathit{Occurs}_\eta(y,v)))\\
&\forall x{:}\rho\forall y{:}\eta.(\mathit{Occurs}_\eta(y,f_{[r]}(x)) \leftrightarrow \mathit{Occurs}_\eta(y,x))\\
&\forall u{:}\rho\forall v{:}\eta\forall y{:}\eta.(\mathit{Occurs}_\eta(y,f_@(v,u)) \leftrightarrow (y = v \vee \mathit{Occurs}_\eta(y,u)))
\end{aligned}$$

Figure 4.5: $\Sigma_{\mathit{Occurs}_\eta}$, many-sorted theory of nominal occurrence in a formula.

is easy to overcome (e.g., using a Peano-like finite encoding of the infinite constants $c_p, c_q, \ldots$ or introducing a new subsort of ρ for propositions) so we will not worry about this.

Finally, one can encode rule (BG$_r$) in a similar way:

$$\begin{aligned}\forall x{:}\rho\forall yz{:}\eta.((y \neq z \wedge \neg\mathit{Occurs}_\eta(z,x) \wedge \mathrm{Thm}(f_@(y,f_{\langle r\rangle}(z)))) \rightarrow&\\ \mathrm{Thm}(f_@(z,f_{[r]}(x))))&\end{aligned} \tag{4.16}$$

where $f_{\langle r\rangle}(t)$ is a shortcut for $f_\neg(f_{[r]}(f_\neg(t)))$. We can now put all the pieces together.

Definition 4.4 ($\Gamma_{\mathcal{H}(@)}$)**.** We define, $\Gamma_{\mathcal{H}(@)}$, the first-order theory of derivability in the system of Figure 4.3 as the minimum set such that:

1. it contains all the formulas of $\Sigma_{\mathit{Occurs}_\eta}$,
2. if φ is an axiom of the system of Figure 4.3, then it contains the well-sorted universal closure of Thm($\varphi^\star$),
3. it contains the encoding of the inference rules (Name) and (BG$_r$), namely, formulas (4.15) and (4.16),
4. it contains the following formulas (encoding the axiom schemes (MP), (Nec$_r$) and (Nec$_@$)):

$$\begin{aligned}
&\forall x{:}\rho\forall y{:}\rho.((\mathrm{Thm}(f_\rightarrow(x,y)) \wedge \mathrm{Thm}(x)) \rightarrow \mathrm{Thm}(y))\\
&\forall x{:}\rho.(\mathrm{Thm}(x) \rightarrow \mathrm{Thm}(f_{[r]}(x)))\\
&\forall x{:}\rho\forall y{:}\eta.(\mathrm{Thm}(x) \rightarrow \mathrm{Thm}(f_@(y,x)))
\end{aligned}$$

It is not hard to see that this encoding indeed works as expected.

Theorem 4.3. *For every formula φ of $\mathcal{H}(@)$, we have $\vdash \varphi$ iff $\Gamma_{\mathcal{H}(@)} \models_{\mathrm{FO}} \varphi^*$.*

We close this chapter discussing the modifications that need to be done to handle an axiomatization for $\mathcal{H}(@,\downarrow)$.

It is customary to present hybrid logics containing the $\downarrow$ operator making a syntactical distinction between bindable and non-bindable nominals. The former are usually called *world variables* or *state variables* and the latter simply *nominals*. For simplicity, we have preferred to avoid such distinction, so far. The systems of Blackburn and ten Cate [2006] assume there is indeed such distinction; so we need a way to overcome this slight asymmetry. We will achieve this by assuming defined a countable infinite set $\mathsf{Var} \subseteq \mathsf{Nom}$ such that $\mathsf{Nom} \setminus \mathsf{Var}$ is countably infinite. Observe this assumption is compatible with Convention 1 so there is no generality lost.

Blackburn and ten Cate [2006] show that one can obtain a complete axiomatization for $\mathcal{H}(@,\downarrow)$ by taking the one for $\mathcal{H}(@)$ we have considered so far and "just" adding an *axiom scheme* (i.e., an infinite, recursive, set of new axioms). Actually, the notion of substitution has to be modified as well to account for *safe* substitution of nominals by variables and of variables by variables. Here, by "safe" we mean that accidental captures of variables must be avoided. While this is mentioned only casually in that paper, it has some impact from the point of view of encoding theoremhood: we need to add inference rules to handle various forms of *non-uniform* substitution and, unlike uniform substitution, these rules require a non-trivial encoding.

Figure 4.6 displays the rules of this system. It introduces the axiom scheme (DA) and two new (non-uniform) substitution rules, (Subst$_{it}$) and (Subst$_{st}$). We have introduced some slight modifications (in the form of side conditions) with respect to the system of Blackburn and ten Cate [2006] in order to avoid the derivation of theorems that do not conform to Convention 1; incidentally, this allows us to formulate (safe) instantiations of variables by nominals in terms of direct substitutions of nominals.

Let us pay some attention to the axiom scheme (DA). It introduces an instance of formula $@_a(\downarrow s.\varphi \leftrightarrow \varphi(s/a))$ for each bindable nominal s and each formula φ, as long as s does not occur bound in φ. Here a is a fixed nominal the belongs to $\mathsf{Nom} \setminus \mathsf{Var}$. Notice that the side-condition, guarantees that $\downarrow s.\varphi$ conforms to Convention 1 and also that in the substitution $\varphi(s/a)$, a cannot be accidentally captured.

The additional substitution rules are similar to (Subst$_{ij}$); they differ in the side-conditions and in whether they involve nominals in Var or $\mathsf{Nom}\setminus\mathsf{Var}$. Rule (Subst$_{st}$), for example, replaces an arbitrary variable s by an arbitrary variable t in a theorem φ, as long as t does not occur at all in φ. This side-condition guarantees that Convention 1 is preserved which, in turn, prevents accidental captures. This rule allows both for the substitution of both free and bound variables in φ. Rule (Subst$_{it}$), on the other hand, substitutes a nominal i by a free variable t in a theorem φ; t must not occur bound in φ

Axioms

(KN1)	$\vdash p \to (p \wedge p)$	
(KN2)	$\vdash (p \wedge q) \to p$	
(KN3)	$\vdash (p \to q) \to \neg(q \wedge r) \to \neg(p \wedge r)$	
(K_r)	$\vdash [r](p \to q) \to [r]p \to [r]q$	
($\mathsf{K}_@$)	$\vdash @_a(p \to q) \to @_a p \to @_a q$	
($\text{SD}_@$)	$\vdash @_a p \leftrightarrow \neg @_a \neg p$	
($\text{Ref}_@$)	$\vdash @_a a$	
(Agree)	$\vdash @_a @_b p \leftrightarrow @_b p$	
(Intro)	$\vdash a \to (p \leftrightarrow @_a p)$	
(Back_r)	$\vdash \langle r \rangle @_a p \to @_a p$	

Axiom schemes

(DA)	$\vdash @_a(\downarrow s.\varphi \leftrightarrow \varphi(s/a))$	$(s \notin \mathit{Bnd}(\varphi))$

Inference rules

(MP)	If $\vdash \varphi \to \psi$ and $\vdash \varphi$ then $\vdash \psi$	
(Nec_r)	If $\vdash \varphi$ then $\vdash [r]\varphi$	
($\text{Nec}_@$)	If $\vdash \varphi$ then $\vdash @_a\varphi$	
(Subst)	If $\vdash \varphi$ then $\vdash \varphi\sigma$	
(Subst_{ij})	If $\vdash \varphi$ then $\vdash \varphi(i/j)$	
(Subst_{it})	If $\vdash \varphi$ then $\vdash \varphi(i/t)$	$(t \notin \mathit{Bnd}(\varphi))$
(Subst_{st})	If $\vdash \varphi$ then $\vdash \varphi(s/t)$	$(t \notin \mathit{Bnd}(\varphi) \cup \mathit{Free}(\varphi))$
(Name)	If $\vdash @_i\varphi$ then $\vdash \varphi$	$(i \notin \mathit{Free}(\varphi))$
(BG_r)	If $\vdash @_i\langle r \rangle j$ then $\vdash @_j[r]\varphi$	$(j \neq i$ and $j \notin \mathit{Free}(\varphi))$

Notes:

- $\langle r \rangle\varphi$ is short for $\neg[r]\neg\varphi$.
- $\varphi \leftrightarrow \psi$ is short for $(\varphi \to \psi) \wedge (\psi \to \varphi)$.
- a and b are fixed nominals such that $a \notin \mathsf{Var}$ and $b \notin \mathsf{Var}$.
- i and j are arbitrary nominals such that $i \notin \mathsf{Var}$ and $j \notin \mathsf{Var}$.
- s and t are arbitrary nominals such that $s \in \mathsf{Var}$ and $t \in \mathsf{Var}$.

Figure 4.6: Hilbert-style system for the hybrid logic $\mathcal{H}(@, \downarrow)$.

in order to conform to Convention 1.

In order to encode this system, we first need to accommodate the additional constructs in the hybrid language by expanding the first-order vocabulary with a new sort declaration $\nu \sqsubseteq \rho$, constants $c_s, c_t, \ldots$ of sort ν, and function symbols $f'_{@} : \nu \times \rho \to \rho$ and $f_{\downarrow} : \nu \times \rho \to \rho$. Bijection $(\cdot)^*$ of Definition 4.4 must be extended with these additional clauses, for $s \in \mathsf{Var}$[2]:

$$(s)^* \overset{def}{=} c_s$$
$$(@_s\varphi)^* \overset{def}{=} f'_{@}(c_s, \varphi^*)$$
$$(\downarrow s.\varphi)^* \overset{def}{=} f_{\downarrow}(c_s, \varphi^*)$$

In a similar way, we must extend theory Γ_{Occurs_η} of Figure 4.5 to account for the new terms by adding these formulas:

$$\forall x{:}\nu \forall y{:}\eta.(\neg Occurs_\eta(y, x)) \tag{4.17}$$
$$\forall u{:}\rho \forall v{:}\nu \forall y{:}\eta.(Occurs_\eta(y, f'_{@}(v, u)) \leftrightarrow Occurs_\eta(y, u)) \tag{4.18}$$
$$\forall u{:}\rho \forall v{:}\nu \forall y{:}\eta.(Occurs_\eta(y, f_{\downarrow}(v, u)) \leftrightarrow Occurs_\eta(y, u)) \tag{4.19}$$

Let us now consider the encoding of axiom scheme (DA). The first thing to observe is that, just like (Subst) and (Subst$_{ij}$), it involves a substitution. However, we can't just use the same technique in this case: not only we have a side-condition to check, but we now need to build a new formula with occurrences of both the subformula φ before and after the replacement. The way to deal with this scheme is with the following formula:

$$\begin{aligned}\forall x{:}\rho \forall y{:}\eta \forall z{:}\nu.(\neg IsBnd(z,x) \to \\ \mathrm{Thm}(f_{@}(y, f_{\leftrightarrow}(f_{\downarrow}(z, x), rpl^{\nu}_{\eta}(x, z, y)))))\end{aligned} \tag{4.20}$$

where $f_{\leftrightarrow}(x, y)$ is a shorthand for $f_{\wedge}(f_{\to}(x, y), f_{\to}(y, x))$ and where we would like to interpret function symbol $rpl^{\nu}_{\eta} : \rho \times \nu \times \eta \to \rho$ as "replacement inside an arbitrary formula (sort ρ) of the free occurrences of a variable (sort ν) by a non-bindable nominal (sort η)" and predicate $IsBnd : \nu \times \rho$ as "the variable (sort ν) occurs bound in a formula".

Using equality, it is possible to define first-order theories that gives the desired meaning to rpl^{ν}_{η} and $IsBnd$. Figure 4.7, for example, shows a theory for rpl^{ν}_{η}, which we call $\Sigma_{rpl^{\nu}_{\eta}}$. A theory Σ_{IsBnd} for $IsBnd$ is analogous to Σ_{Occurs_η} so we will skip its details.

We finally turn to the new substitution rules. Rule (Subst$_{it}$) resembles axiom scheme (DA) in many ways, so it is not surprising we can encode it in a similar way. We need an additional function symbol $rpl^{\eta}_{\nu} : \rho \times \eta \times \nu \to \rho$ that, intuitively, represents the uniform replacement of a non-bindable nominal (sort η) by a variable (sort ν) in an arbitrary formula.

$$\forall x{:}\rho \forall y{:}\eta \forall z{:}\nu.((\neg IsBnd(z, x) \wedge \mathrm{Thm}(x)) \to \mathrm{Thm}(rpl^{\eta}_{\nu}(x, y, z))))) \tag{4.21}$$

[2]There is no need to extend also $(\cdot)^*$ since no axiom contains variables.

$$\forall y{:}\nu\forall z{:}\eta\forall x{:}\eta.(rpl^{\nu}_{\eta}(x,y,z) = x)$$
$$\forall y{:}\nu\forall z{:}\eta\forall x{:}\nu.(x = y \rightarrow rpl^{\nu}_{\eta}(x,y,z) = z)$$
$$\forall y{:}\nu\forall z{:}\eta\forall x{:}\nu.(x \neq y \rightarrow rpl^{\nu}_{\eta}(x,y,z) = x)$$
$$\forall y{:}\nu\forall z{:}\eta.(rpl^{\nu}_{\eta}(c_p,y,z) = c_p), \text{ for every constant } c_p \text{ of sort } \rho$$
$$\forall y{:}\nu\forall z{:}\eta\forall x{:}\rho.(rpl^{\nu}_{\eta}(f_{\neg}(x),y,z) = f_{\neg}(rpl^{\nu}_{\eta}(x,y,z)))$$
$$\forall y{:}\nu\forall z{:}\eta\forall x{:}\rho.(rpl^{\nu}_{\eta}(f_{[r]}(x),y,z) = f_{[r]}(rpl^{\nu}_{\eta}(x,y,z)))$$
$$\forall y{:}\nu\forall z{:}\eta\forall u{:}\rho\forall v{:}\rho.(rpl^{\nu}_{\eta}(f_{\wedge}(u,v),y,z) = f_{\wedge}(rpl^{\nu}_{\eta}(u,y,z),rpl^{\nu}_{\eta}(v,y,z)))$$
$$\forall y{:}\nu\forall z{:}\eta\forall u{:}\rho\forall v{:}\rho.(rpl^{\nu}_{\eta}(f_{\rightarrow}(u,v),y,z) = f_{\rightarrow}(rpl^{\nu}_{\eta}(u,y,z),rpl^{\nu}_{\eta}(v,y,z)))$$
$$\forall y{:}\nu\forall z{:}\eta\forall x{:}\rho\forall u{:}\eta.(rpl^{\nu}_{\eta}(f_{@}(u,x),y,z) = f_{@}(u,rpl^{\nu}_{\eta}(x,y,z)))$$
$$\forall y{:}\nu\forall z{:}\eta\forall u{:}\nu\forall x{:}\rho.(u = y \rightarrow rpl^{\nu}_{\eta}(f'_{@}(u,x),y,z) = f_{@}(z,rpl^{\nu}_{\eta}(x,y,z)))$$
$$\forall y{:}\nu\forall z{:}\eta\forall u{:}\nu\forall x{:}\rho.(u \neq y \rightarrow rpl^{\nu}_{\eta}(f'_{@}(u,x),y,z) = f'_{@}(u,rpl^{\nu}_{\eta}(x,y,z)))$$
$$\forall y{:}\nu\forall z{:}\eta\forall u{:}\nu\forall x{:}\rho.(u = y \rightarrow rpl^{\nu}_{\eta}(f_{\downarrow}(u,x),y,z) = f_{\downarrow}(u,x))$$
$$\forall y{:}\nu\forall z{:}\eta\forall u{:}\nu\forall x{:}\rho.(u \neq y \rightarrow rpl^{\nu}_{\eta}(f_{\downarrow}(u,x),y,z) = f_{\downarrow}(u,rpl^{\nu}_{\eta}(x,y,z)))$$

Figure 4.7: $\Sigma_{rpl^{\nu}_{\eta}}$, a many-sorted, first-order theory of instantiation.

Similarly, in order to encode rule (Subst$_{st}$) we will rely on a function symbol $rpl^{\nu}_{\nu} : \rho \times \nu \times \nu \rightarrow \rho$ that represents uniform substitution of bindable nominals and on a predicate symbol $Occurs_{\nu} : \nu \times \rho$ that expresses the concept "a bindable nominal occurs in a formula". The idea is analogous to that of (4.21):

$$\forall x{:}\rho\forall y{:}\nu\forall z{:}\nu.((\neg Occurs_{\nu}(z,x) \wedge \mathrm{Thm}(x)) \rightarrow \mathrm{Thm}(rpl^{\nu}_{\nu}(x,y,z))))) \quad (4.22)$$

We leave to the reader the details of defining the theories $\Sigma_{rpl^{\eta}_{\nu}}$, $\Sigma_{rpl^{\nu}_{\nu}}$ and $\Sigma_{Occurs_{\nu}}$ that give the desired meaning to rpl^{η}_{ν}, rpl^{ν}_{ν} and $Occurs_{\nu}$, respectively. The first two are analogous to $\Sigma_{rpl^{\nu}_{\eta}}$ while the last one is similar to $\Sigma_{Occurs_{\eta}}$. We can now put all the pieces together.

Definition 4.5 ($\Gamma_{\mathcal{H}(@,\downarrow)}$)**.** We define, $\Gamma_{\mathcal{H}(@,\downarrow)}$, the first-order theory of derivability in the system of Figure 4.6 as the extension of $\Gamma_{\mathcal{H}(@)}$ with:

1. formulas (4.17), (4.18) and (4.19), to extend $\Sigma_{Occurs_{\eta}}$,
2. all the formulas in $\Sigma_{rpl^{\nu}_{\eta}}$, $\Sigma_{rpl^{\eta}_{\nu}}$, $\Sigma_{rpl^{\nu}_{\nu}}$, Σ_{IsBnd} and $\Sigma_{Occurs_{\nu}}$,
3. the encoding of axiom scheme (DA), and inference rules (Subst$_{it}$) and (Subst$_{st}$), namely, formulas (4.20), (4.21), and (4.22).

Theorem 4.4. *For φ of $\mathcal{H}(@,\downarrow)$, we have $\vdash \varphi$ iff $\Gamma_{\mathcal{H}(@,\downarrow)} \models_{\mathrm{FO}} \varphi^*$.*

Blackburn and ten Cate [2006] observe that in the case of $\mathcal{H}(@,\downarrow)$ the non-orthodox inference rules (Name) and ($\mathrm{BG_r}$) may be replaced by:

Axioms

($\mathrm{BG}_{r\downarrow}$) $\vdash @_a[r]{\downarrow}x.@_a\langle r\rangle x$ ($x \in \mathsf{Var}$ is fixed)

Axiom schemes

($\mathrm{Name}_\downarrow$) $\vdash {\downarrow}s.(s \to \varphi) \to \varphi$ ($s \notin \mathit{Bnd}(\varphi) \cup \mathit{Free}(\varphi)$)

Inference rules

($\mathrm{Nec}_\downarrow$) If $\vdash \varphi$ then $\vdash {\downarrow}s.\varphi$

One can easily adapt the encoding we developed so far in order to handle these rules instead.

The authors also observe that, even though in this case all the inference rules are orthodox, axiom schemes ($\mathrm{Name}_\downarrow$) and (DA) are still somewhat displeasing from an algebraic point of view. We may add that the non-uniform substitution rules (Subst_{it}) and (Subst_{st}), not explicitly mentioned in their presentation, seem rather unorthodox too.

Notice that in the encodings we developed in this chapter, the need for equality came always from rules of this kind. Blackburn and ten Cate [2006] conjecture that $\mathcal{H}(@,\downarrow)$ cannot be axiomatized using a finite number of orthodox axiom schemes and inference rules. If this wouldn't be the case, then we would be able to encode them in first-order logic without equality; which seems to be strong evidence of the correctness of the conjecture.

In this chapter, we have seen how to adapt the well-known first-order encoding of modal Hilbert-style systems to the hybrid case. In the process, we identified to unorthodox rules and axiom schemes as responsibles for the introduction of equality in the first-order language. This kind of encodings are known to have poor computational behaviour, but if reasoning over some non-elementary classes of models is required, they might constitute the only choice. In the next two chapters, we will investigate translations based on semantic intuitions which, in general, are better behaved and appropriate for applications.

Chapter 5

Relational translations

One can usually see a logic from either a semantic or syntactical point of view. In the former, one uses rules that determine what it means for a formula to be *true* under a suitable interpretation, for the latter one characterizes what constitutes a formally valid chain of deductions. We have already seen modal and hybrid logics from both points of view: in Chapter 1 we defined the semantics of both $\mathcal{ML}$ and $\mathcal{H}(@, \downarrow)$ while in Chapter 4 we considered Hilbert-style axiomatic systems that describe valid reasonings for both logics. Moreover, we saw in the latter that the valid proofs characterized by these systems can also be described in first-order logic. Similarly, it is possible to encode in first-order logic the semantics of modal and hybrid logics we saw in Chapter 1; this will be the subject of this chapter.

We will see that the encoding of modal and hybrid semantics (customarily know as the *standard translations*) is quite simple and straightforward. This is related to the fact that modal and hybrid models are no more than first-order models in disguise.

Simple as the standard translations may be, when employed as a tool in translation-based automated reasoning, they tend to give poor results. This is a well-known fact. The so-called *layered translation* is a variation of the standard translation for the basic modal logic that was proposed to help a resolution-based first-order theorem prover avoid some of the pitfalls it usually run into when processing formulas generated by the standard translation. We will review this translation as well as its motivations, and we will finish this chapter developing a layered translation for $\mathcal{H}(@, \downarrow)$.

5.1 The standard translation

A Kripke model can be alternatively seen as a standard first-order model over an appropriate first-order language: every $p \in \mathsf{Prop}$ is seen as a one-place predicate symbol while every $r \in \mathsf{Rel}$ is considered a two-place predicate symbol. We shall call this the *correspondence language of* $\mathcal{ML}$. Thus, we can identify a Kripke model $\mathcal{M} = \langle W, R, V\rangle$ with the first-order interpretation $\mathcal{I}^{\mathcal{M}} = \langle W, \cdot^{\mathcal{I}^{\mathcal{M}}}\rangle$ such that:

$$\begin{aligned} r^{\mathcal{I}^{\mathcal{M}}} &= R(r) \\ p^{\mathcal{I}^{\mathcal{M}}} &= V(p). \end{aligned}$$

In practice, we won't normally make the distinction between $\mathcal{M}$ and $\mathcal{I}^{\mathcal{M}}$ and simply use $\mathcal{M}$ also as a first-order interpretation.

Using this correspondence language, it is straightforward to encode in first-order logic the semantic clauses of Definition 1.10. This leads to the so-called *standard translation* of modal formulas.

Definition 5.1 (ST_x)**.** The standard translation to first-order logic ST_x maps basic modal formulas into first-order logic formulas in the correspondence language with a free variable x as follows:

$$\begin{aligned} ST_x(p) &\overset{def}{=} p(x) \\ ST_x(\neg\varphi) &\overset{def}{=} \neg ST_x(\varphi) \\ ST_x(\varphi \wedge \psi) &\overset{def}{=} ST_x(\varphi) \wedge ST_x(\psi) \\ ST_x([r]\varphi) &\overset{def}{=} \forall y.r(x,y) \rightarrow ST_y(\varphi),\ y \text{ is fresh} \end{aligned}$$

Theorem 5.1. *Let φ be a modal formula. Then the following hold:*

1. *For every $\mathcal{M}$ and $w \in |\mathcal{M}|$, we have $\mathcal{M}, w \models_K \varphi$ iff $\mathcal{M} \models_{\text{FO}} ST_x(\varphi)[w]$.*
2. *$\models_K \varphi$ iff $\models_{\text{FO}} \forall x.ST_x(\varphi)$.*
3. *φ is satisfiable iff $\exists x.ST_x(\varphi)$ is satisfiable.*

Gabbay [1981] observed that we actually only need two variables to carry out the standard translation:

$$\begin{aligned} ST_x([r]\varphi) &\overset{def}{=} \forall y.r(x,y) \rightarrow ST_y(\varphi) \\ ST_y([r]\varphi) &\overset{def}{=} \forall x.r(y,x) \rightarrow ST_x(\varphi) \end{aligned}$$

Since the fragment of first-order logic with two variables is known to be decidable [Scott, 1962, Mortimer, 1975] using this version of the standard translation one can have (at least theoretically) a decision method for $\mathcal{ML}$, unlike the encoding of Chapter 4. It was later observed that even without restricting the number of variables used, the standard translation also falls into a decidable fragment: the so called *guarded fragment* of first-order logic [Andréka et al., 1998].

The decidability of these fragments by itself, of course, does not guarantee that a complete, off-the-shelf theorem prover will include strategies to decide them. In fact, as of today most don't. But even if we can't expect to simply compose a translation to first-order logic with an automated theorem prover and obtain a decision procedure, it does make sense (and may be even more important from a practical point of view) to try to obtain good "average" performance out of this lightweight approach. As it will be discussed next, it is known that this is actually not the case with the plain standard translation, and smarter translations are required.

But before moving on, it should be mentioned that there are indeed some known procedures based on first-order logic resolution for deciding the aforementioned fragments. For example, de Nivelle and Pratt-Hartmann [2001], show how to decide the function-free[1] two-variable fragment containing equality using a resolution-based procedure. Their technique is not trivial to implement, though: an initial set of clauses has to be saturated, then a renaming phase takes place, and finally the resulting clause set has to be saturated again. Ganzinger and de Nivelle [1999], on the other hand, prove that the function-free guarded fragment with equality can be decided using superposition and standard term-orderings provided a special clausification method is employed. A similar method was later employed by de Nivelle and de Rijke [2003] to show that the function-free guarded fragment without equality is decidable using plain resolution. However, we are not aware of any published performance evaluation of any of these methods. Moreover, the lack of support for constants and/or equality would make them unsuitable for handling the first-order translation of hybrid logics.

Simple experiments have shown that even for unsatisfiable formulas (where because of refutational completeness the prover shall, in theory, terminate), the performance of the standard translation may seriously lag behind procedures that have been purpose-built for modal reasoning (see, e.g., [Ohlbach et al., 2001]). Two proposals have been put forward to overcome this; one exploits the so-called *tree-model property* of many modal logics while the other takes advantage of an alternative to relational semantics based on the use of functions instead of relations. We devote the rest of this chapter to introduce the first approach and discuss its extension to hybrid logics; the same will be done for the second method in the following chapter.

5.2 The layered translation

Areces et al. [2000] identified one aspect of the relatively poor performance of modern first-order theorem provers on the standard translation of modal formulas and proposed a way to improve it. In order to understand both their analysis and solution, we first need to introduce some classical concepts.

[1] "Function-free" in this thesis will always imply "constant-free".

A binary relation $T \subseteq A \times A$ is called a *tree* if its transitive closure T^+ is total and irreflexive, and its converse T^{-1} is a partial function defined for all but one $r \in A$; such r is called the *root* of T. We say that a Kripke model $\mathcal{M} = \langle W, R, V \rangle$ is a *tree-model with root w* if $\bigcup_{r \in \mathsf{Rel}} R(r)$ is a tree with root w. Finally, we say that a logic possesses the *tree-model property* whenever every satisfiable formula is satisfiable in a tree-model.

Proposition 5.1. *The basic modal logic has the tree-model property.*

The tree-model property is a classical tool used, for example, in many decidability results (see, e.g., [Blackburn et al., 2002], which also includes a proof of Proposition 5.1). In fact, it has been identified by some authors as a key property in explaining the *robust decidability*[2] of modal logics [Vardi, 1996, Grädel, 2001].

Areces et al. [2000] used the tree-model property to identify one of the reasons why resolution-based automated theorem provers perform badly on the standard translation of modal formulas, at least when compared to modal theorem provers. The following is an illustrative example.

Example 5.1 (Areces et al. [2000])**.** Consider the following formula:

$$[r](p \to \langle r \rangle p) \tag{5.1}$$

Needless to say, it is trivially satisfiable. But let's take our modal logician's hat off for a moment, and put us in the place of a "mechanical", first-order theorem prover, instead.

By Theorem 5.1, formula (5.1) is satisfiable if and only if the following formula is satisfiable:

$$\exists x \forall y. \big(r(x,y) \to \big(p(y) \to \exists z.(r(y,z) \land p(z))\big)\big) \tag{5.2}$$

And this formula is equivalent, in terms of satisfiability, to:

$$\forall y. \big(\big(r(c,y) \to (p(y) \to r(y, f(y)))\big) \land \big(r(c,y) \to (p(y) \to p(f(y)))\big)\big) \tag{5.3}$$

where f and c are introduced by skolemization. Notice that (5.3) is in clause normal form and, therefore, processable by a resolution-based first-order theorem prover. The aim of such a prover would be to show that the following clauses are simultaneously satisfiable:

$$\neg p(u) \lor \neg r(c,u) \lor p(f(u)) \tag{5.4}$$
$$\neg p(v) \lor \neg r(c,v) \lor r(y, f(v)) \tag{5.5}$$

Here we are following the standard convention in which variables in different clauses must be disjoint (in this case, u and v). Since we have put ourselves

$$\begin{array}{l}
1{:}\neg p(u) \vee \neg r(c, f^0(u)) \vee p(f^1(u))\\
2{:}\neg p(v) \vee \neg r(c, f^0(v)) \vee r(f^0(v), f^1(v))\\
3{:}\neg p(w) \vee \neg r(c, f^1(w)) \vee r(f^1(w), f^2(w)) \vee \neg r(c, f^0(w))\\
4{:}\neg p(x) \vee \neg r(c, f^2(x)) \vee r(f^2(x), f^3(x)) \vee \neg r(c, f^1(x)) \vee \neg r(c, f^0(x))\\
5{:}\neg p(y) \vee \neg r(c, f^3(y)) \vee r(f^3(y), f^4(y)) \vee \neg r(c, f^2(y)) \vee \neg r(c, f^1(y)) \vee \neg r(c, f^0(y))\\
\qquad\qquad\qquad\qquad\qquad\qquad\vdots
\end{array}$$

where $f^0(t) \overset{def}{=} t$ and $f^{n+1}(t) \overset{def}{=} f^n(t)$

Figure 5.1: For $n > 0$, clause $n + 2$ is obtained by resolving on p between clauses $n + 1$ and 1 using $\{z_{n+1} \mapsto f(z_{n+2}), z_1 \mapsto z_{n+2}\}$ as unifier, where z_i is the unique free variable in clause i.

in the place of a resolution-based prover, we have to mechanically saturate this set, which quickly leads us into trouble, as shown in Figure 5.1.

In order to highlight the pattern in the infinite derivation, we have opted to use the $f^n(t)$ notation, even preferring $f^0(t)$ over t in some cases. With this in mind, it is easy to recognize in clauses 1 and 2 formulas (5.4) and (5.5) respectively. Also for the sake of symmetry, Figure 5.1 omits the following ground clause, that results from resolving 1 and 2 on r using as unifier $\{u \mapsto f(c), v \mapsto c\}$:

$$3' : \neg p(c) \vee \neg p(f^1(c)) \vee \neg p(f^2(c)) \vee \neg r(c, f^0(c))$$

The reader may verify that no other inference is possible. Moreover, none of these infinitely many clauses is redundant (in the technical sense of the word, cf. Chapter 2) and can be deleted.

Time to put our modal logician's hat on again. The first thing to observe is that for a formula as trivially satisfiable as (5.1) it is far from appropriate to require a large number of inferences to determine it, not to say an infinite number of them. In order to better understand what went wrong in Example 5.1, let us take a finer look at the inference that lead to clause 3 in Figure 5.1.

The unifier used in this inference was $\{v \mapsto f(w), u \mapsto w\}$, which means we are considering the case $v \approx f(w)$ and $u \approx w$, which implies, $v \approx f(u)$. Now, recall that in (5.4) and (5.5), variables u and v where introduced only to fulfill the requirement of variables in different clauses to be disjoint; but they both stand for the universally quantified variable y in (5.3). Therefore, the case considered in this inference is, in essence, that $y \approx f(y)$ in (5.3).

[2]By "robust" it is meant that it remains decidable even under very expressive extensions, such as transitive closure and fixed-point operators.

So far, so good. What does it mean in a model for (5.3) the requirement that every y and $f(y)$ denote the same element? The term $f(y)$ was introduced as the skolemization of the existentially quantified variable z occurring in (5.2) and z was introduced to witness an r-successor of x during translation of (5.1).

Summing up, this inference is considering the case where the successor of universally quantified variable y is y itself. But such a model, even if it satisfies (5.2), would not be a tree-model and, because of Proposition 5.1, may be left "unexplored" without compromising completeness. In other words, because of the tree-model property, it is safe to block the generation of clause 3 (and, therefore, all the infinitely many clauses derived after it) altogether.

The layered translation achieves this by generating the following initial clauses instead of (5.4) and (5.5):

$$\neg p_1(u) \vee \neg r_0(c, u) \vee p_2(f(u)) \tag{5.6}$$

$$\neg p_1(v) \vee \neg r_0(c, v) \vee r_1(y, f(v)) \tag{5.7}$$

where r_0, r_1, p_1 and p_2 are distinct predicate symbols. There are clearly no resolvents for this set of clauses. But where do these subindices come from? Observe that these clauses correspond to the standard translation of:

$$[r_0](p_1 \rightarrow \langle r_1 \rangle p_2) \tag{5.8}$$

which is simply (5.1) with every relation and proposition symbol annotated with its modal depth. The intuition is that any tree-model for (5.1), e.g. the one in Figure 5.2a, can be turned into a model for (5.8) by simply replacing every proposition symbol in a node with a version annotated with the distance from the node to the root and partitioning the interpretation of relation r into interpretations for $r_0, r_1 \ldots$ depending on the distance to the root of the source node in the relation. This is shown in Figure 5.2b.

In short, tree-models naturally induce a notion of *layers* while the syntax and semantics of modal logic allows us to reify this layering in the formula in a satisfiability-preserving way. This explicit layering, in turn, constraints the search space of a resolution-based first-order theorem prover.

Time to formalize this translation. We assume a fixed modal signature $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Rel} \rangle$ and define $\mathcal{S}_{\times\omega} = \langle \mathsf{Prop} \times \omega, \mathsf{Rel} \times \omega \rangle$. For every $p \in \mathsf{Prop}$, $r \in \mathsf{Rel}$ and $i \in \omega$ we will typically write p_i and r_i for $p \times i$ and $r \times i$ respectively.

Definition 5.2 (LT_x)**.** The layered translation to first-order logic, LT_x, that maps basic modal formulas of signature $\mathcal{S}$ into first-order logic formulas with a free variable x in the correspondence language for $\mathcal{S}_{\times\omega}$ is defined as

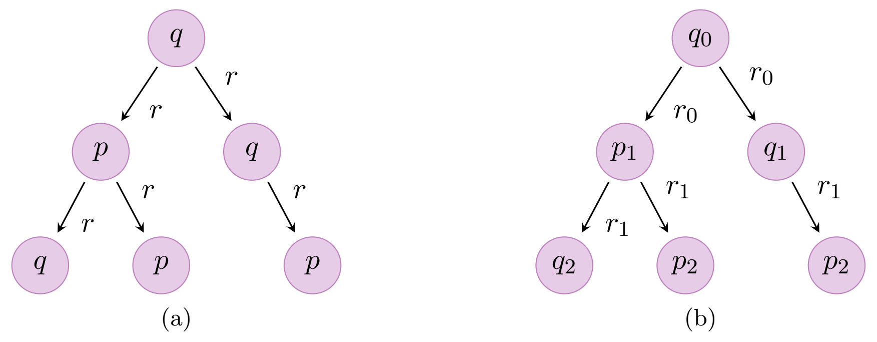

Figure 5.2: Tree-model (b) is obtained from (a) by "annotating" proposition and relation symbols with the distance to the root of the node where they occur.

$LT_x(\varphi) = LT^0_x(\varphi)$ where:

$$\begin{aligned} LT^n_x(p) &\overset{def}{=} p_n(x) \\ LT^n_x(\neg\varphi) &\overset{def}{=} \neg LT^n_x(\varphi) \\ LT^n_x(\varphi \wedge \psi) &\overset{def}{=} LT^n_x(\varphi) \wedge LT^n_x(\psi) \\ LT^n_x([r]\varphi) &\overset{def}{=} \forall y. r_n(x, y) \rightarrow LT^{n+1}_y(\varphi),\ y \text{ is fresh} \end{aligned}$$

Theorem 5.2. *Let φ be a modal formula. Then the following hold:*

1. $\models_K \varphi$ *iff* $\models_{\mathrm{FO}} \forall x. LT_x(\varphi)$.
2. φ *is satisfiable iff* $\exists x. LT_x(\varphi)$ *is satisfiable.*

Of course, the layered translation only preserves satisfiability, but this is enough for translation-based automated reasoning. Areces et al. [2000] report on experiments comparing the standard and layered translations, where the latter consistently outperforms the former for up to four orders of magnitude.

In the case of multi-modal logics (i.e., those where $|\mathsf{Rel}| > 1$) we can still make a further optimization to the translation. Figure 5.3a is a tree-model for a modal logic with at least two relations r and s. Now, observe the two leaves labeled with p: one is a node reachable from the root by first taking an r-relation and then an s-relation, while the second one requires first taking an s-relation followed by an r-relation. Let us refer to them as the rs-successor and sr-successor of the root, respectively. Just like in the analysis of Example 5.1, if we would like the first-order prover to avoid considering the case where a node is both an rs-successor and a sr-successor.

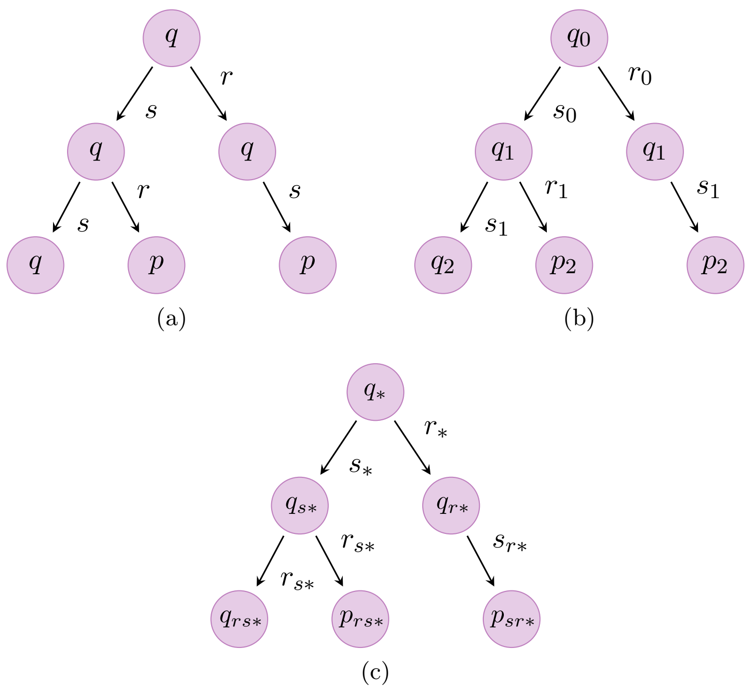

Figure 5.3: Tree-model (b) is obtained from (a) by annotating all symbols with distances to the root (cf. Fig. 5.3b), while (c) is annotated with "paths" instead of distances.

Unfortunately, we cannot rely on layering as per Definition 5.2 to do this: as Figure 5.3b illustrates, modal depth as the only form of annotation is not enough to syntactically distinguish an sr-successor from an rs-successor.

The solution is to use *paths expressions* instead of distances as annotations. Figure 5.3c illustrates the idea. "Path expressions" are simply sequences of symbols in Rel (plus a new symbol $*$ reserved for the root) and represent a path from the node back to the root. For example, the path expression $*$ means we are already at the root, $r*$ means taking an r-relation (backwards) takes you to the root and $sr*$ mean "to reach the root first go through an s-relation and then through an r-relation". Adapting Definition 5.2 to use path expressions instead of distances is trivial.

5.3 A layered translations for hybrid logics

We now turn to hybrid logics. Here too, hybrid models can be seen as first-order models over a correspondence language that extends the one for the basic modal logic with the equality relation and a constant i for every

$i \in \mathsf{Nom}$. Although we will make a distinction between first-order constants and variables (e.g., we will distinguish ground and non-ground terms), we will sometimes find convenient to treat constants as variables; most typically we will use constants as argument of $\forall$ and $\exists$. In this particular case, we will simply assume $\forall c.\varphi(c)$ (with c a constant) to be equivalent to $\forall x_c.\varphi(x_c)$ for some fresh variable x_c.

Definition 5.3 ($ST_x^{\mathcal{H}}$)**.** The standard hybrid translation $ST_x^{\mathcal{H}}$ maps hybrid formulas into first-order logic formulas in the correspondence language with (at most) a free variable x by extending ST_x with the following clauses:

$$\begin{aligned} ST_x^{\mathcal{H}}(i) &\overset{def}{=} (x = i) \\ ST_x^{\mathcal{H}}(@_i\varphi) &\overset{def}{=} ST_i^{\mathcal{H}}(\varphi) \\ ST_x^{\mathcal{H}}(\downarrow i.\varphi) &\overset{def}{=} \exists i.(i = x \wedge ST_x^{\mathcal{H}}(\varphi)) \end{aligned}$$

Observe that for every nominal i, $ST_i^{\mathcal{H}}(\varphi)$ is a ground formula. Similarly, even for a variable x, $ST_x^{\mathcal{H}}(@_i\varphi)$ is ground.

Theorem 5.3. *Let φ be a hybrid formula. Then the following hold:*

1. *For every $\mathcal{M}$ and $w \in |\mathcal{M}|$, we have $\mathcal{M}, w \models \varphi$ iff $\mathcal{M} \models_{\text{FO}} ST_x^{\mathcal{H}}(\varphi)[w]$.*
2. *$\models \varphi$ iff $\models_{\text{FO}} \forall x.ST_x^{\mathcal{H}}(\varphi)$.*
3. *φ is satisfiable iff $\exists x.ST_x^{\mathcal{H}}(\varphi)$ is satisfiable.*

For every basic modal formula φ, $ST_x^{\mathcal{H}}(\varphi) = ST_x(\varphi)$ holds. In particular, we have $ST_x^{\mathcal{H}}\big([r](p \to \langle r\rangle p)\big) = ST_x\big([r](p \to \langle r\rangle p)\big)$ which shows that for translation-based reasoning, $ST_x^{\mathcal{H}}$ must suffer of the same performance problems ST_x does. Unfortunately, as soon as we have nominals in the language we lose the tree-model property. For example, observe that no tree-model can satisfy $i \wedge \langle r\rangle i$ since the node that satisfies i would have to be an r-successor of itself. But the tree-model property played a key role in the correctness of the layered translation LT_x.

In the rest of this section we will exploit weaker forms of the tree-model property in order to have a layered translation for the hybrid setting. But before proceeding, we should verify that there are indeed (proper) hybrid formulas that exhibit a behavior similar to that of Example 5.1.

Example 5.2. Consider the following trivially satisfiable formula:

$$@_i[r]\neg p \wedge @_j[r](p \to \langle r\rangle p) \tag{5.9}$$

By applying $ST_x^{\mathcal{H}}$ and then transforming to clausal form we obtain the following formula:

$$\forall y.\big(r(i,y) \to \neg p(y)\big) \wedge \forall y.\big(r(j,y) \wedge p(y) \to \exists z.(r(y,z) \wedge p(z))\big) \tag{5.10}$$

$$\begin{array}{l}
1 : \neg p(u) \vee \neg r(j, f^0(u)) \vee p(f^1(u)) \\
2 : \neg p(v) \vee \neg r(i, f^0(v)) \\
3 : \neg p(w) \vee \neg r(i, f^1(w)) \vee \neg r(j, f^0(w)) \\
4 : \neg p(x) \vee \neg r(i, f^2(x)) \vee \neg r(j, f^0(x)) \vee \neg r(j, f^1(x)) \\
5 : \neg p(y) \vee \neg r(i, f^3(y)) \vee \neg r(j, f^0(y)) \vee \neg r(j, f^1(y)) \vee \neg r(j, f^2(y)) \\
6 : \neg p(z) \vee \neg r(i, f^4(z)) \vee \neg r(j, f^0(z)) \vee \neg r(j, f^1(z)) \vee \neg r(j, f^2(z)) \vee r(j, f^3(z)) \\
\qquad\qquad\qquad\qquad\vdots
\end{array}$$

Figure 5.4: For $n > 0$, clause $n + 2$ is obtained by resolving on p between clauses $n + 1$ and 1 using $\{z_{n+1} \mapsto f(z_{n+2}), z_1 \mapsto z_{n+2}\}$ as unifier, where z_i is the unique free variable in clause i.

From which, by introducing a function symbol f to skolemize z, we obtain the following clause set, suitable for a resolution-based theorem-prover:

$$\neg p(u) \vee \neg r(j, u) \vee p(f(u)) \tag{5.11}$$

$$\neg p(v) \vee \neg r(i, v) \tag{5.12}$$

$$\neg p(w) \vee \neg r(j, w) \vee r(w, f(w)) \tag{5.13}$$

Figure 5.4 shows an infinite derivation by resolution from this set of clauses. In fact, notice only formulas (5.11) and (5.12) are used (they correspond to clauses 1 and 2). Moreover, (5.13) may also resolve against every clause in Figure 5.4 but clause 1, leading in every case to a ground clause.

Just like in Example 5.1, we want to argue that the inference that lead to clause 3 is irrelevant. For this we will first introduce the $\mathcal{H}(@)$ counterpart of the tree-model property and use it to analyze this example and guide us towards a suitable layered translation. Unsurprisingly, the property we are about to introduce will not hold for $\mathcal{H}(@, \downarrow)$ and we will have to review the developed translation in that case.

In the basic modal logic there is only one point of evaluation; in $\mathcal{H}(@)$, using the @ operator we can have several simultaneous such points. This already suggests we have to move from trees to *forests*.

Definition 5.4 (Forest-models)**.** Let $\mathcal{M} = \langle W, R, V, g\rangle$ be a hybrid model. We say $\mathcal{M}$ is a *forest-model* whenever there exists a collection of tree-models $\mathcal{M}_k = \langle W_k, R_k, V_k\rangle$ such that all the W_k are pairwise disjoint, $W = \bigcup_i W_i$, $R(r) = \bigcup R_i(r)$ for all r, $V(p) = \bigcup V_i(p)$ for all p, and for every nominal i there exists a k such that $g(i)$ is the root of $\mathcal{M}_k$.

Alternatively, we say $\mathcal{M}$ is a *quasi-forest-model* if $\mathcal{M}' = \langle W, R', V, g\rangle$, with $R'(r) = \{(w, v) \mid (w, v) \in R(r) \text{ and } g(i) \neq v, \forall i\}$, is a forest-model.

Forest-models are, intuitively, a collection of trees such that nominals only name nodes that are a root; a quasi-forest-model, on the other hand,

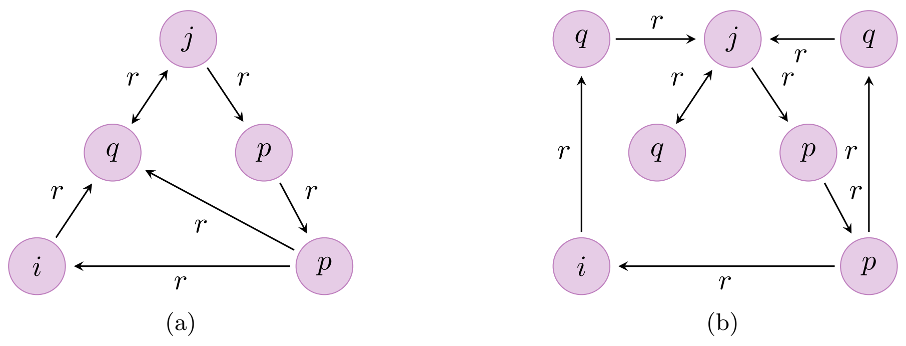

Figure 5.5: Models (a) and (b) are $\mathcal{H}(@)$-equivalent but (b) is a quasi-forest-model.

is turned into a proper forest-model by removing every arrow arriving at a named node. Figure 5.5 shows two models that satisfy (5.9). None of them is a forest-model, although model (b) is indeed a quasi-forest-model.

A closer look at Figure 5.5 shows these two models are related in other ways. First, they are in fact bisimilar models, which implies they coincide on every @-formula of $\mathcal{H}(@)$. Second, it is possible to derive (b) from (a) using a construction similar to classical *unravelings* [Blackburn et al., 2002]. In fact, it is easy to show that one can "unravel" any hybrid model into a bisimilar one that is also a quasi-forest-model. From this, one gets the following result.

Proposition 5.2 (Quasi-forest-model property)**.** *Let φ be a satisfiable formula of $\mathcal{H}(@)$, then φ is satisfiable in a quasi-forest-model. Moreover, if nominals occur in φ only as first arguments of @, then φ is satisfiable in a forest-model.*

So we can now take a new look at Example 5.2 in the light of Proposition 5.3. The first thing to observe is that when considering the satisfiability of (5.9) we can restrict our attention to forest-models. We can use this fact to show that deriving clause 3 is not necessary. Observe that the inference that leads to this clause uses $\{u \mapsto w, v \mapsto f(w)\}$ as unifier. An analysis analogous to the one performed for Example 5.1 shows that, in essence, this unifier covers the case where there is some node reachable both from i and from j; since this is not possible in a forest-model, it is safe to ignore it altogether.

We saw that for the case of the basic modal logic one can annotate each relation and predicate symbol with a path-expression, in order to avoid unifications between terms that can be assumed to denote nodes on different paths on a tree-model. In the case of $\mathcal{H}(@)$-formulas that are satisfiable in a

1:	$\neg p_{*_i}(i)$	
2:	$r_{*_j}(j, c)$	
3:	$c = i$	
4:	$p_{r*_j}(c)$	
5:	$\neg p_{*_i}(c)$	(by Paramodulation on 1 and 3)
6:	$r_{*_j}(j, i)$	(by Paramodulation on 2 and 3)
7:	$p_{*_j}(i)$	(by Paramodulation on 4 and 3)

Figure 5.6: No contradiction is derived from clauses (5.18)–(5.21).

forest-model, we can use the same idea in order to avoid unifications between terms we know may be assumed to denote elements on different trees. We simply need to extend path expressions with a $*_i$ symbol for every nominal i, to account for the fact that nominals are roots of trees in forest-models.

The trivial extension of Definition 5.2 to $\mathcal{H}(@)$ formulas where nominals occur only as first argument of @ would produce, for formula (5.9), the following clauses, from which no inference can be drawn:

$$\neg p_{r*_j}(u) \lor \neg r_{*_j}(j, u) \lor p_{rr*_j}(f(u)) \tag{5.14}$$

$$\neg p_{r*_i}(v) \lor \neg r_{*_i}(i, v) \tag{5.15}$$

$$\neg p_{r*_j}(w) \lor \neg r_{*_j}(j, w) \lor r_{r*_j}(y, f(w)) \tag{5.16}$$

This approach is not correct for arbitrary $\mathcal{H}(@)$-formulas. though. To see this consider the following formula:

$$@_i \neg p \land @_j \langle r \rangle (i \land p) \tag{5.17}$$

It is clearly an unsatisfiable formula and, therefore, we expect to be able to derive a contradiction from the following clauses (here c is a skolem constant):

$$\neg p_{*_i}(i) \tag{5.18}$$

$$r_{*_j}(j, c) \tag{5.19}$$

$$c = i \tag{5.20}$$

$$p_{r*_j}(c) \tag{5.21}$$

However, as shown in Figure 5.6, no contradiction can be derived from these clauses. Let us try to see what the problem is. According to the path expression of clauses 2 and 4, c is expected to be “reachable” from a tree with root j. But from clause 3, since c must coincide with the node named i, we conclude that c is also “reachable” from i and, therefore, predicates referring to c should also be annotated with $*_i$. That is, from clauses 4 and 3 in Figure 5.6 we would like to derive not only 7, but also:

$$7' : p_{*_i}(c)$$

With this clause it would be trivial to derive a contradiction.

In a way, the problem comes from the fact that, unlike the case of forest-models, on quasi-forest-models, there may be more than one path-expression applicable to a node. For example, in the model of Figure 5.5b, we can assign to node i the expressions $*_i$ and $rrr*_j$.

The way out of this problem is simply to encode path expressions as ground first-order terms which become an additional parameter of the predicates. We illustrate the idea by giving the resulting clause set for (5.17). Here $*_i$ and $*_j$ will be constants and $\dot{r}$ a unary function:

$$\neg p(*_i, i) \tag{5.22}$$

$$r(*_j, j, c) \tag{5.23}$$

$$c = i \tag{5.24}$$

$$*_i = \dot{r}(*_j) \tag{5.25}$$

$$p(\dot{r}(*_j), c) \tag{5.26}$$

It is interesting to compare these formulas with (5.18)–(5.21). Not only the path expressions are now first-order terms, but we have an additional formula stating that path expressions $*_i$ and $\dot{r}(*_j)$ match, meaning that there may be elements reachable following both kind of paths.

Time to formalize this translation. First, the correspondence language will extend that of the standard translation with a constant $*_i$ for every nominal i, and a unary function symbol $\dot{r}$ for every $r \in \mathsf{Rel}$. We will use an additional constant $*$ for the (anonymous) point of evaluation. Path expressions are encoded as first-order terms using exclusively these new symbols. Such terms will be called p-terms. Second, the arity of all relational symbols is increased by one to accommodate the new parameter used for path expressions. To simplify the definitions and subsequent proofs, we will work with formulas in negation normal form for the rest of this chapter.

Definition 5.5 ($LT^{\mathcal{H}}$)**.** For any $\mathcal{H}(@)$-formula φ in negation normal form the layered translation $LT^{\mathcal{H}}_x(\varphi)$ is defined as $LT^{\mathcal{H}}_{x,*}(\varphi)$ where

$$\begin{aligned}
LT^{\mathcal{H}}_{x,t}(i) &\overset{def}{=} x = i \wedge *_i = t \\
LT^{\mathcal{H}}_{x,t}(\neg i) &\overset{def}{=} x \neq i \\
LT^{\mathcal{H}}_{x,t}(p) &\overset{def}{=} p(t, x) \\
LT^{\mathcal{H}}_{x,t}(\neg p) &\overset{def}{=} \neg p(t, x) \\
LT^{\mathcal{H}}_{x,t}(\varphi \wedge \psi) &\overset{def}{=} LT^{\mathcal{H}}_{x,t}(\varphi) \wedge LT^{\mathcal{H}}_{x,t}(\psi) \\
LT^{\mathcal{H}}_{x,t}(\varphi \vee \psi) &\overset{def}{=} LT^{\mathcal{H}}_{x,t}(\varphi) \vee LT^{\mathcal{H}}_{x,t}(\psi) \\
LT^{\mathcal{H}}_{x,t}(\langle r \rangle \varphi) &\overset{def}{=} \exists y . r(t, x, y) \wedge LT^{\mathcal{H}}_{y,\dot{r}(t)}(\varphi), \ y \text{ is fresh}
\end{aligned}$$

$$LT^{\mathcal{H}}_{x,t}([r]\varphi) \overset{def}{=} \forall y.r(t,x,y) \rightarrow LT^{\mathcal{H}}_{y,\dot{r}(t)}(\varphi),\ y \text{ is fresh}$$
$$LT^{\mathcal{H}}_{x,t}(@_i\varphi) \overset{def}{=} LT^{\mathcal{H}}_{i,*_i}(\varphi)$$

Theorem 5.4. *For every $\mathcal{H}(@)$-formula φ in negation normal form, and every p-term t, $LT^{\mathcal{H}}_{x,t}(\varphi)$ is satisfiable iff φ is satisfiable.*

Proof. For the right-to-left implication, observe that equality of p-terms occurs only positively in $LT^{\mathcal{H}}_{x,t}(\varphi)$ and, thus, any model for φ is trivially turned into a model for $LT^{\mathcal{H}}_{x,t}(\varphi)$ by making all the p-terms denote the same domain element.

For the other direction, let $\mathcal{I} = \langle \mathcal{D}, \cdot^{\mathcal{I}} \rangle$ be such that $\mathcal{I} \models_{\text{FO}} LT^{\mathcal{H}}_{x,t}(\varphi)[a]$ for some $a \in \mathcal{D}$. We define the hybrid model $\mathcal{M}_{\mathcal{I}} = \langle W, R, V, g \rangle$ where

$$\begin{aligned}
W &\overset{def}{=} \mathcal{D} \times \mathcal{D}\\
R(r) &\overset{def}{=} \{\big((t^{\mathcal{I}}, x), (\dot{r}(t)^{\mathcal{I}}, y)\big) \mid t \text{ is a } p\text{-term and } (t^{\mathcal{I}}, x, y) \in r^{\mathcal{I}}\}\\
V(p) &\overset{def}{=} \{(t^{\mathcal{I}}, x) \mid t \text{ is a } p\text{-term and } (t^{\mathcal{I}}, x) \in p^{\mathcal{I}}\}\\
g(i) &\overset{def}{=} ({*_i}^{\mathcal{I}}, i^{\mathcal{I}})
\end{aligned}$$

We show by induction on φ that it must be the case $\mathcal{M}_{\mathcal{I}}, (t^{\mathcal{I}}, a) \models \varphi$. We prove only the relevant cases:

Case $\varphi \equiv i$. If $\mathcal{I} \models_{\text{FO}} LT^{\mathcal{H}}_{x,t}(i)[a]$ then $\mathcal{I} \models (x = i \wedge t = *_i)[a]$ and, therefore, $i^{\mathcal{I}} = a$ and $t^{\mathcal{I}} = {*_i}^{\mathcal{I}}$. But this means that $(t^{\mathcal{I}}, a) = ({*_i}^{\mathcal{I}}, i^{\mathcal{I}}) = g(i)$ and, thus, $\mathcal{M}_{\mathcal{I}}, (t^{\mathcal{I}}, i) \models i$.

Case $\varphi \equiv p$. We know that $\mathcal{I} \models_{\text{FO}} p(t,x)[a]$, that is $(t^{\mathcal{I}}, a) \in p^{\mathcal{I}}$. Consequently, $\mathcal{M}_{\mathcal{I}}, (t^{\mathcal{I}}, a) \models p$.

Case $\varphi \equiv \langle r \rangle \psi$. Since $\mathcal{I} \models_{\text{FO}} \exists y.r(t,x,y) \wedge LT^{\mathcal{H}}_{y,\dot{r}(t)}(\psi)[a]$, then there must exist some $b \in \mathcal{D}$ such that $(t^{\mathcal{I}}, a, b) \in r^{\mathcal{I}}$ and $\mathcal{I} \models_{\text{FO}} LT^{\mathcal{H}}_{y,\dot{r}(t)}(\psi)[b]$. But $(t^{\mathcal{I}}, a, b) \in r^{\mathcal{I}}$ implies $\big((t^{\mathcal{I}}, a), (\dot{r}(t)^{\mathcal{I}}, b)\big) \in R(r)$ and by inductive hypothesis $\mathcal{M}_{\mathcal{I}}, (\dot{r}(t)^{\mathcal{I}}, b) \models \psi$. Thus, we have $\mathcal{M}_{\mathcal{I}}, (t^{\mathcal{I}}, a) \models \langle r \rangle \psi$.

Case $\varphi \equiv @_i\psi$. From $\mathcal{I} \models_{\text{FO}} LT^{\mathcal{H}}_{x,t}(@_i\psi)[a]$ we conclude $\mathcal{I} \models_{\text{FO}} LT^{\mathcal{H}}_{i,*_i}(\psi)[a]$ but since this is a ground formula, it must also be the case that $\mathcal{I} \models_{\text{FO}} LT^{\mathcal{H}}_{x,*_i}[i^{\mathcal{I}}]$. Then, by inductive hypothesis, we may conclude that $\mathcal{M}_{\mathcal{I}}, ({*_i}^{\mathcal{I}}, i^{\mathcal{I}}) \models \psi$. Finally, since by definition $g(i) = ({*_i}^{\mathcal{I}}, i^{\mathcal{I}})$, we get $\mathcal{M}_{\mathcal{I}}, (t^{\mathcal{I}}, a) \models @_i\psi$.

$\square$

Corollary 5.1. *For all $\mathcal{H}(@)$-formula φ, φ is satisfiable iff $\exists x.LT^{\mathcal{H}}_x(\varphi)$ is satisfiable.*

Once we move to $\mathcal{H}(@, \downarrow)$ the quasi-forest-model property no longer holds. Intuitively, since $\downarrow$ can be used to "name worlds on-the-fly", a satisfiable formula may force relations between unnamed worlds to occur. Consider, for example the following satisfiable formula:

$$\langle r\rangle p \wedge \langle r\rangle \neg p \wedge [r]\langle r\rangle p \wedge [r]\langle r\rangle \neg p \wedge [r]{\downarrow} i.[r]\langle r\rangle i \tag{5.27}$$

It is not hard to see that in any model $\mathcal{M}$ that satisfies (5.27) there must be at least two nodes with fan-in greater than one. But if we restrict to a signature with only one nominal i, then only one of these nodes may be labeled with i and, therefore, $\mathcal{M}$ cannot be a quasi-forest-model.

Nevertheless, this will not be a problem in extending the layered translation to $\mathcal{H}(@, \downarrow)$. The reader may have already noticed that in the proof of Theorem 5.4, the quasi-forest-model property was not used at all. This property was more a guiding principle that lead us to the formulation of the translation than a theoretical justification for its correctness. In fact, the same can be said about the tree-model property and the layered translation for $\mathcal{ML}$: one can proof the correctness of the former without employing the latter.

With this in mind, it is not hard to extend Definition 5.5 to handle $\downarrow$ in a sound way. The intuition is that whenever a satisfiable formula φ has no quasi-forest-model it is because certain occurrences of ${\downarrow} i$ in φ are forcing some otherwise unnamed nodes to have greater fan-in. But this can be easily expressed in terms of p-terms:

$$LT^{\mathcal{H}}_{x,t}({\downarrow} i.\psi) \overset{def}{=} *_i = t \wedge \exists i.(i = x \wedge LT^{\mathcal{H}}_{x,t}(\psi)) \tag{5.28}$$

The proof of correctness is similar. It is interesting to observe that with this translation, the nominals occurring bound in φ may influence the performance of a resolution-based prover on $LT^{\mathcal{H}}(\varphi)$.

To see this, suppose that φ and φ' only differ in the choice of bound nominals, and that in φ some nominal is bound more than once while in φ' none does. It is easy to see that the set of clauses obtained from $LT^{\mathcal{H}}(\varphi)$ and $LT^{\mathcal{H}}(\varphi')$ will only differ on path expressions; moreover in the set corresponding to $LT^{\mathcal{H}}(\varphi)$ there may be path expressions occurring more often than in the one for $LT^{\mathcal{H}}(\varphi')$, but the reciprocal is not true. The bottom line is: every valid inference in $LT^{\mathcal{H}}(\varphi')$ will be a valid inference in $LT^{\mathcal{H}}(\varphi)$, but there may be inferences in $LT^{\mathcal{H}}(\varphi)$ that are not valid in $LT^{\mathcal{H}}(\varphi')$ because a mismatch of path expressions. Therefore, in order to maximize this kind of blocking, one should restrict to formulas where nominals are bound only once.

Chapter 6

Functional translations

In the late 1980's and early 1990's the functional translation approach appeared simultaneously and independently in a number of publications (see, e.g. [Ohlbach, 1988b,a, Fariñas del Cerro and Herzig, 1988, Zamov, 1989, Auffray and Enjalbert, 1989, 1992]). Based on an alternative view of relational structures, this translation produces formulas that can be much more compact and with a more shallow term structure than those obtained with the standard translation. These are good properties from a practical point of view. Moreover, just like the layered translation, the functional translation avoids many of the pitfalls of the standard translation. Although it is harder to understand the correctness of this translation when compared to those of Chapter 5, we will later see that the hybrid case blends in quite naturally.

The functional translation is often presented using a many-sorted logic, like the one employed in Chapter 4. While this simplifies the presentation, it also hides part of the complexity of the resulting formula. In Section 6.4 we will investigate conditions under which sort annotations can be safely removed from a translated formula without changing its satisfiability status.

6.1 Functional models, functional translation

The key to understand the functional translation is via an alternative representation of relational structures. To simplify this presentation, we will discuss relational structures with only one relation and later generalize this to the multi-relational case. Consider, then, Figure 6.1a which shows a relational structure with domain $W = \{a, b, c\}$ and one relation r. One can alternatively represent this structure using three total functions f, g and h, and a predicate de, as long as the following property holds:

$$\forall xy.\big(r(x,y) \leftrightarrow (\neg de(x) \wedge (f(x) = y \vee g(x) = y \vee h(x) = y))\big) \qquad (6.1)$$

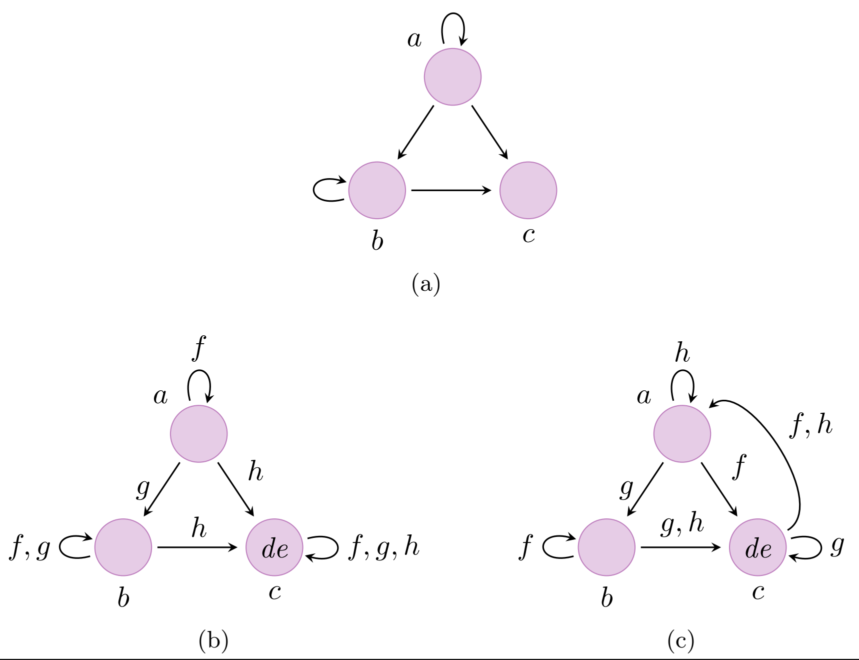

Figure 6.1: A relational structure (a) is represented in (b) and (c) using total functions f, g, h and a predicate de.

The idea is to use de (for "dead end") to "mark" those states that have no r-successor, and to use f, g and h on each state to "witness" each r-successor. There are many valid arrangements for f, g and h; Figures 6.1b, and c show two such representations. It is straightforward to verify they both satisfy condition (6.1).

Proposition 6.1. *Let $\mathcal{I}$ be a finite first-order interpretation for a language with a two-place relation symbol r. Then there exists an interpretation $\mathcal{I}'$ extending $\mathcal{I}$ to a language that additionally contains a one-place relation symbol de and unary function symbols $f_1, f_2 \ldots f_n$, such that:*

$$\mathcal{I}' \models_{\text{FO}} \forall xy. \big(r(x, y) \leftrightarrow (\neg de(x) \wedge (f_1(x) = y \vee f_2(x) = y \ldots \vee f_n(x) = y))\big)$$

But functions $f_1, f_2, \ldots f_n$ can be alternatively represented using only one binary function f, that takes a *function index* as an additional argument. This is easily expressed in a language with two sorts ω and ι, the former will refer to proper nodes of the model, the latter to function indices.

Proposition 6.2. *Let $\mathcal{I}$ be a first-order interpretation for a language with a relation symbol of sort $r : \omega \times \omega$. Then there exists an interpretation $\mathcal{I}'$ extending $\mathcal{I}$ to a language that additionally contains a relation symbol $de : \omega$*

and a function symbol $f : \iota \times \omega \to \omega$, such that:

$$\forall xy{:}\omega.\big(r(x,y) \leftrightarrow (\neg de(x) \wedge \exists z{:}\iota. f(z,x) = y)\big)$$

Observe that, unlike Proposition 6.1, this encoding is suitable for infinite interpretations too.

We now have everything in place to define a notion of functional Kripke model. Observe that we will be generalizing the previous propositions to the multi-relational case in the obvious way.

Definition 6.1 (Functional Kripke models)**.** A functional Kripke model for a modal signature $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Rel} \rangle$ is a many-sorted interpretation

$$\langle W, I, \{f_r \mid r \in \mathsf{Rel}\}, \{de_r \mid r \in \mathsf{Rel}\}, V \rangle$$

where W and I are non-empty domains for sorts ω and ι, respectively; $V : \mathsf{Prop} \to 2^W$; and, for each $r \in \mathsf{Rel}$, $f_r : I \times W \to W$ and $de_r : 2^W$.

Clearly, every functional Kripke model induces a (relational) Kripke model such that for every relation r the following holds:

$$\forall x, y{:}\omega. r(x,y) \leftrightarrow (\neg de_r(x) \wedge \exists z{:}\iota. f(z,x) = y) \tag{6.2}$$

Therefore, we say that a functional model $\mathcal{M}_{\mathcal{F}}$ satisfies a modal formula φ if and only if its induced relational model satisfies φ. We will later be interested in *maximal models*, that is, functional models where every possible function is realized.

Definition 6.2 (Maximal models)**.** Consider a functional Kripke model $\mathcal{I} = \langle W, I, \{f_r \mid r \in \mathsf{Rel}\}, \{de_r \mid r \in \mathsf{Rel}\}, V \rangle$ and let $r^{\mathcal{I}}$ be the relation induced by (6.2) for each $r \in \mathsf{Rel}$. We say $\mathcal{I}$ is *maximal* if for each total function $\gamma : W \to W$ such that $(w, \gamma(w)) \in r^{\mathcal{I}}$ for all $w \in W$, there exists an $i \in I$ for which $f_r(i, x) = \gamma(x)$, for all x.

Any functional model can trivially be extended to a maximal one.

Proposition 6.3. *A formula of the basic modal logic is satisfiable iff there exists a (maximal) functional Kripke model that satisfies it.*

Just like relational Kripke models can be seen also as first-order models in the correspondence language, it is clear that functional Kripke models are first-order models in the *functional (sorted) correspondence language*: we use two sorts ω and ι, each de_r and every $p \in \mathsf{Prop}$ are seen as one-place predicate symbols of sort ω, while there is a binary function symbol $f_r : \iota \times \omega \to \omega$ for every $r \in \mathsf{Rel}$. Finally, we can see the *functional translation* simply as a "standard" translation to many-sorted first-order logic over the functional correspondence language.

Definition 6.3 (FT)**.** The functional translation to first-order logic FT_x that maps basic modal formulas into first-order logic formulas in the functional correspondence language with a free variable x of sort ω is defined as $FT_x(\varphi) = FT'_x(\varphi)$ where, for every term t:

$$\begin{aligned}
FT'_t(p) &\overset{def}{=} p(t)\\
FT'_t(\neg\varphi) &\overset{def}{=} \neg FT'_t(\varphi)\\
FT'_t(\varphi \wedge \psi) &\overset{def}{=} FT'_t(\varphi) \wedge FT'_t(\psi)\\
FT'_t([r]\varphi) &\overset{def}{=} \neg de_r(t) \to \forall z{:}\iota. FT'_{f_r(z,t)}(\varphi),\ z \text{ is fresh}
\end{aligned}$$

Theorem 6.1. *Let φ be a formula of $\mathcal{ML}$. Then the following hold:*

1. $\models_K \varphi$ *iff* $\models_{\text{FO}} \forall x{:}\omega. FT_x(\varphi)$.
2. φ *is satisfiable iff* $\exists x{:}\omega. FT_x(\varphi)$ *is satisfiable.*

In fact, it is straightforward to see that Theorem 6.1 can be extended to work also in the case where we are reasoning with respect to a first-order definable class of frames. For example, suppose we require r to be interpreted as a transitive relation, which is expressible in first-order logic as:

$$\forall xyz.(r(x,y) \wedge r(y,z) \to r(x,z)) \tag{6.3}$$

By combining it with the equivalence (6.2) and doing some valid transformations we obtain the functional equivalent:

$$\forall x{:}\omega.(\neg de(x) \to \forall ab{:}\iota \exists c{:}\iota. f_r(b, f_r(a,x)) = f_r(c,x)) \tag{6.4}$$

Finally, formula φ of $\mathcal{ML}$ will be satisfiable in the class of models where r is transitive if and only if the conjunction of (6.4) and $FT(\varphi)$ is satisfiable.

Of course, this is also true in the case of the standard (relational) translation. Notice, however, that for the layered translation discussed in Chapter 5 it is not clear in general how to express frame conditions without interfering with the layering.

One of the major appeals of the functional translations is that there are some interesting "optimizations" that can be done on it, in particular one that guarantees termination in the modal case when any modern resolution-based first-order theorem prover is employed. This will be discussed in Section 6.3, but before that we will introduce a suitable functional translation for hybrid logics.

6.2 Functional translation for hybrid logics

Let us recapitulate for a minute. In the functional translation, objects of the domain are represented by functional terms. These terms are of the form:

$$f_{r_n}(y_n, f_{r_{n-1}}(y_{n-1}, \ldots f_{r_1}(y_1, x)\ldots)) \tag{6.5}$$

Here x represents the initial point-of-evaluation, while $y_1, \ldots y_n$ are "indices" of functions. Intuitively, this term is denoting an element that is reachable from x by first following r_1, then r_2, etc. and the index y_i is encoding which is the r_i-successor in each case.

But basically this means we can extend the functional translation to hybrid logics in a way similar to what was done for the standard translation: simply use first-order constants to represent nominals. Then the first-order term i will denote the element of the domain where the nominal i holds, while $f_r(y, i)$ will denote the r-successor of i indexed by y.

Formally, we will then extend the functional correspondence language with a constant i of sort ω for each $i \in \mathsf{Nom}$, and define a functional hybrid model in the expected way.

Definition 6.4 (Hybrid functional models)**.** A hybrid functional model for a modal signature $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel} \rangle$ is a structure

$$\langle W, I, \{f_r \mid r \in \mathsf{Rel}\}, \{de_r \mid r \in \mathsf{Rel}\}, V, g\rangle$$

where W and I are non-empty domains for sorts ω and ι, respectively; $V : \mathsf{Prop} \to 2^W$; $g : \mathsf{Nom} \to W$ and, for each $r \in \mathsf{Rel}$, $f_r : I \times W \to W$ and $de_r : 2^W$.

Again, every hybrid functional model induces a hybrid model. The hybrid functional translation is finally defined as follows.

Definition 6.5 ($FT^{\mathcal{H}}$)**.** The hybrid functional translation to first-order logic $FT^{\mathcal{H}}_x$, that maps $\mathcal{H}(@, \downarrow)$-formulas into first-order logic formulas in the functional correspondence language with a free variable x of sort ω, is defined as $FT^{\mathcal{H}}_x(\varphi) = FT^{\mathcal{H}'}_x(\varphi)$ where$FT^{\mathcal{H}'}_t$ extends FT'_t with the following clauses:

$$\begin{aligned}
FT^{\mathcal{H}'}_t(i) &\stackrel{\mathit{def}}{=} (i = t)\\
FT^{\mathcal{H}'}_t(@_i\varphi) &\stackrel{\mathit{def}}{=} FT^{\mathcal{H}'}_i(\varphi)\\
FT^{\mathcal{H}'}_t(\downarrow i.\varphi) &\stackrel{\mathit{def}}{=} \exists i{:}\omega.(i = t \wedge FT^{\mathcal{H}'}_t(\varphi))
\end{aligned}$$

Theorem 6.2. *Let φ be an $\mathcal{H}(@, \downarrow)$-formula. Then the following hold:*

1. $\models \varphi$ *iff* $\models_{\mathrm{FO}} \forall x{:}\omega.FT^{\mathcal{H}}_x(\varphi)$.
2. φ *is satisfiable iff* $\exists x{:}\omega.FT^{\mathcal{H}}_x(\varphi)$ *is satisfiable.*

The proof is straightforward.

6.3 Optimized functional translations

Let us revisit the formula of Example 5.1:

$$[r](p \to \langle r\rangle p) \tag{6.6}$$

By Theorem 6.1, this formula is satisfiable if and only if its functional translation is satisfiable too (the r subscript is omitted):

$$\begin{aligned}&\exists x{:}\omega.\big(\neg de(x) \to \\ &\qquad \forall y{:}\iota.(p(f(y,x)) \to (\neg de(f(y,x)) \land \exists z{:}\iota.p(f(z,f(y,x))))))\big)\end{aligned} \tag{6.7}$$

And by skolemizing x and z with fresh constant c and function g we obtain the equisatisfiable formula:

$$\neg de(c) \to \forall y{:}\iota.\big(p(f(y,c)) \to (\neg de(f(y,c)) \land p(f(g(y),f(y,c))))\big) \tag{6.8}$$

Formula (6.8) contains two skolem symbols: a constant and a unary function. The so-called "optimized functional translation" [Ohlbach and Schmidt, 1997] guarantees that only *constants* need to be introduced during skolemization, and since this is achieved by simply reordering quantifiers, the translated formula is essentially the same. Because skolem functions may cause complex terms to be built up during resolution, the optimized translation may drastically reduce the saturation process. Moreover, this simplifies the development of terminating resolution strategies [Schmidt, 1999].

To illustrate the idea behind the optimized translation, let us consider again formula (6.6). Figure 6.2a shows a model that satisfies (6.6) when evaluated on node w. Figure 6.2b, on the other hand, shows a functional model for (6.8), the functional translation of (6.6). Clearly, using (6.2) this model induces the one of Figure 6.2a.

Observe now that for $x \mapsto w$ there are two possible values for y, namely f and g. If $y \mapsto f$, then we must pick $z \mapsto f$, while for $y \mapsto g$ we must select $z \mapsto g$. Therefore, the right value for z is effectively a function of y, as witnessed by the skolemization. But here comes the interesting part: we can "rearrange" the assignment of functions in a way that makes the choice of z independent of y. An example is shown in Figure 6.2c; this model also induces (a) but here the right choice is $z \mapsto g$ independent of the value of y. Maximal models (cf. Definition 6.2) include all possible "rearrangement" of functions and therefore allow us to prove that, in terms of satisfiability, this can always be assumed.

Ohlbach and Schmidt [1997] take advantage of this observation and prove that it is sound, in terms of satisfiability, to swap two consecutive quantifiers. Therefore one can take a formula obtained using the basic functional translation and simply make all the existential quantifiers come before universal ones, effectively avoiding the introduction of skolem functions.

Proposition 6.4 (Ohlbach and Schmidt [1997])**.** *If φ is a formula in prenex normal form with a quantifier-free matrix φ' and φ is equivalent to the functional translation of a formula of $\mathcal{ML}$, then φ is satisfiable iff $\exists\overline{x}{:}\iota\forall\overline{y}{:}\iota\varphi'$ is satisfiable too, where all the $\overline{x}$ and $\overline{y}$ are existentially and universally quantified, respectively, in φ.*

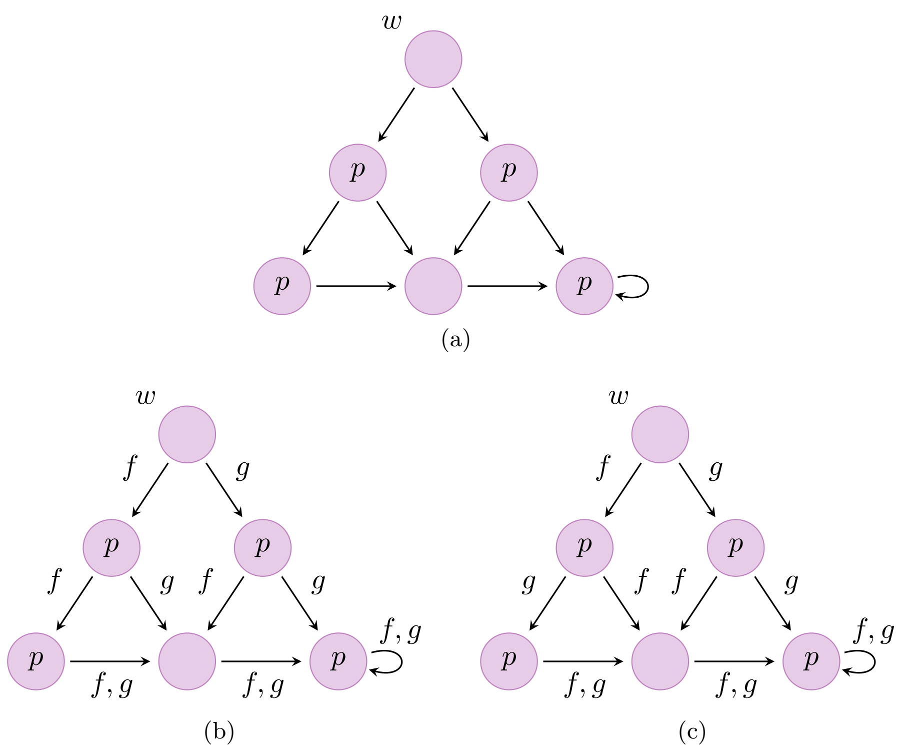

Figure 6.2: A model (a) for formula (6.6) and two models (b) and (c) for its functional translation

We will follow the proof given by Ohlbach and Schmidt [1997] to verify that the result also holds in the hybrid case. This proof will be later reused in Section 6.4 to exhibit certain conditions under which all sort annotations can simply be removed.

Ohlbach [1988b] exhibits a syntactic invariant for the functionally translated terms, which he calls "prefix stability" (known also as "unique path property" [Auffray and Enjalbert, 1992]). Intuitively, what this property says is that we can build a tree (or a forest) out of the set of terms and subterms occurring in a formula such that: i) nodes of the tree are terms, ii) arcs are labeled with variables of sort ι, iii) t_1 is the father of t_2 using an arc labeled by y iff $t_2 = f_r(y, t_1)$ for some $r \in \mathsf{Rel}$ and iv) every variable of sort ι labels only one arc.

Definition 6.6 (Prefix stability)**.** We say a formula φ is *prefix-stable* if, given the set T_φ of all terms occurring in φ, it holds that for every variable y of sort ι in T_φ, there exist a term t and an $r \in \mathsf{Rel}$ such that if y occurs in a term in T_φ, then every occurrence of y is of the form $f_r(y, t)$. We will call $f_r(y, t)$ the *context of y in φ*.

As an example, consider the variable y that occurs in the functional translation of (6.6): all its occurrences have the same context, namely, $f(y,x)$. It is straightforward to see that this property follows from the way functional terms are built in the translation and, in particular, it holds in the hybrid case too.

This syntactic property has something to say from a semantic point of view. Again, suppose all occurrences of a variable y of sort ι are in the context $f_r(y,t)$ for a fixed t. Then, for any given model, the function "indexed" by y will be *relevant* only to determine successors of (the interpretation of) t. This is formally expressed in the following lemma.

Lemma 6.1. *Let φ be a prefix-stable formula in the functional correspondence language and let y be a free variable in φ that occurs in context $f_r(y,t)$, with all the variables in t free in φ. Furthermore, let $\mathcal{I}$ be a functional model and v a valuation. If i_1 and i_2 are two elements of the domain of $\mathcal{I}$ for sort ι such that $f_r^{\mathcal{I}}(i_1, v(t)) = f_r^{\mathcal{I}}(i_2, v(t))$, where $v(t)$ is the interpretation of term t using $\mathcal{I}$ and v, then*

$$\mathcal{I} \models_{\text{FO}} \varphi[v(y \mapsto i_1)] \textit{ iff } \mathcal{I} \models_{\text{FO}} \varphi[v(y \mapsto i_2)]$$

Proof. The proof goes by induction on φ. We will look only at the base case, so assume φ is of the form $p(t')$, with $f_r(y,t)$ a subterm of t'. Let us define $v_1 = v(y \mapsto i_1)$ and $v_2 = v(y \mapsto i_2)$. The first thing to observe is that because y does not occur in t, we have $v_1(t) = v_2(t) = v(t)$. Therefore, we also have $v_1(f_r(y,t)) = v_2(f_r(y,t))$. Finally, again because of prefix-stability, we know there is no other occurrence of y in t' and, therefore, $v_1(t') = v_2(t')$, from which the expected result follows. An analogous reasoning can be used to handle the case where φ is an equality of the form $t_1 = t_2$. The inductive cases follow simply by inductive hypothesis. □

This lemma is a corrected version of a lemma by Ohlbach and Schmidt [1997]. In *Lemma 4.6* of their paper, variables in t are allowed to occur bound in φ (this is explicitly considered in their proof), but in that case one can build a simple counterexample (as long as the functions indexed by i_1 and i_2 are requested to coincide only on one point of the domain). In any case, using this lemma, one can prove the following result (cf. [Ohlbach and Schmidt, 1997, *Theorem 4.7*]).

Theorem 6.3. *Let φ be a prefix-stable formula in the functional correspondence language and let y be a free variable in φ that occurs in context $f_r(y,t)$, with all the variables in t free in φ. Finally, let $\mathcal{I}$ be a* maximal *functional model. Then, for every valuation v we have:*

$$\mathcal{I} \models_{\text{FO}} \forall x_1 \dots x_k{:}\iota \exists y{:}\iota.\varphi[v] \textit{ iff } \mathcal{I} \models_{\text{FO}} \exists y{:}\iota \forall x_1 \dots x_k{:}\iota.\varphi[v]$$

Proof. The right-to-left implication is already valid in the general case, so we only have to consider the left-to-right one. Suppose, then, that the antecedent holds. This means there must exist some function $\gamma : I^k \to I$ such that, $\mathcal{I} \models_{\text{FO}} \varphi[v(x_1 \mapsto i_1 \dots x_k \mapsto i_k, y \mapsto \gamma(i_1 \dots i_k))]$ holds, for every $i_1 \dots i_k \in I$ (here I is taken to be the interpretation of sort ι). Now, let $g \in I$ be such that, $f_r^{\mathcal{I}}(g, v'(t)) = \gamma(i_1 \dots i_k)$ for $v' = v(x_1 \mapsto i_1 \dots x_k \mapsto i_k)$. Such a g must exist since $\mathcal{I}$ is a maximal model. Therefore, using Lemma 6.1 we may conclude that $\mathcal{I} \models_{\text{FO}} \varphi[v(x_1 \mapsto i_1 \dots x_k \mapsto i_k, y \mapsto g)]$ must hold. But since g is independent of $i_1 \dots i_k$, we finally obtain that $\mathcal{I} \models_{\text{FO}} \exists y{:}\iota \forall x_1 \dots x_k{:}\iota.\varphi[v]$. □

We have everything in place now to define the so-called *optimized functional translation*. The idea is simply to take the standard functional translation and move all existential quantifiers to the front. We will work directly on the hybrid case.

Definition 6.7 ($OFT_x^{\mathcal{H}}$)**.** The hybrid optimized functional translation to first-order logic $OFT_x^{\mathcal{H}}$, that maps $\mathcal{H}(@, \downarrow)$-formulas into first-order logic formulas in the functional correspondence language with a free variable x, is defined as $OFT_x^{\mathcal{H}}(\varphi) = \vartheta(FT^{\mathcal{H}}(\varphi))$, where $\vartheta(\psi)$ takes ψ to prenex normal form and moves all existential quantifiers of sort ι to the front.

Theorem 6.4. *Let φ be an $\mathcal{H}(@, \downarrow)$-formula. Then the following hold:*

1. $\models \varphi$ *iff* $\models_{\text{FO}} \forall x{:}\omega.OFT_x^{\mathcal{H}}(\varphi)$.
2. φ *is satisfiable iff* $\exists x{:}\omega.OFT_x^{\mathcal{H}}(\varphi)$ *is satisfiable.*

Proof. It is enough to prove that $FT_x^{\mathcal{H}}(\varphi)$ is satisfiable iff $\vartheta(FT_x^{\mathcal{H}}(\varphi))$. The right-to-left implication is valid in general. For the, other direction, suppose then that $\mathcal{I} \models_{\text{FO}} FT_x^{\mathcal{H}}(\varphi)[v]$ for some $\mathcal{I}$ and v. Without loss of generality, we may assume that $\mathcal{I}$ is maximal. Let ψ be the result of taking $OFT_x^{\mathcal{H}}(\varphi)$ and moving all existential quantifiers of sort ι after every universal quantifier. Observe that $FT_x^{\mathcal{H}}(\varphi) \to \psi'$ is universally valid, and, therefore, $\mathcal{I} \models_{\text{FO}} \psi[v]$. Now, using Theorem 6.3, we can move every existential quantifier in ψ to the front, one at a time (for there must always exist one such that its bound variable y occurs in a context $f_r(y, t)$ and all the variables in t are either universally quantified or their existential quantifiers have been moved to the top already). This process can be repeated only finitely many times and the resulting formula ψ' satisfies $\mathcal{I} \models_{\text{FO}} \psi'[v]$ and is equivalent to $\vartheta(FT_x^{\mathcal{H}}(\varphi))$. □

Observe that the above proof only requires that a maximal model is always available for a satisfiable formula. But this means that the optimized functional translation also works in case we are interested in satisfiability with respect to a first-order definable frame condition.

Schmidt [1999] shows that if we restrict to the basic modal case, then any refinement of resolution plus the (eagerly applied) condensing rule [Joyner, Jr., 1976] is terminating for the output of the optimized functional translation. Most first-order theorem provers have factoring and subsumption deletion rules, and hence condensing is in fact implicit when the implementation is fair. This means that, in practice, any standard (complete and fair) resolution theorem prover used along with the optimized translation constitutes a decision method. Termination conditions for some frame classes are also investigated. As future work, it would be interesting to see if termination can also be achieved in the case of $\mathcal{H}(@)$.

6.4 Sort erasure

The (optimized) functional translation is typically presented using a many-sorted first-order logic, just like we did so far in this chapter. From a practical point of view, if we are interested in using these translations for automated reasoning, we have roughly two alternatives: i) use a theorem prover based on a calculus designed for the many-sorted first-order logic, ii) simulate sorts in unsorted first-order logic using additional one-place predicate symbols.

There has been work on automated reasoning in the presence of sorts, and in particular in the case of resolution and paramodulation. The basic idea in this case is to use what is called *well-sorted unification*. Walther [1989] gives a brief survey of the area. Weidenbach [2001] describes the implementation of a superposition-based theorem prover with support for sorts. SPASS [Weidenbach et al., 2007] is an advanced theorem prover built on these ideas. Nevertheless, it is rather an exception: most modern first-order theorem provers have no internal support for sorts.

It is argued by Walther [1989] that the simulation of sorts by way of proposition symbols leads to irrelevant inferences. A system like SPASS, avoids these inferences but one has to consider also the additional complexity of well-sorted unification. We may conclude that there is some hidden complexity in the use of sorts that must be taken into account.

What we will show in this section is that in many situations it is safe to simply "erase" sort annotations (that is, without introducing anything else). That this can be done in the case of the basic modal logic (when reasoning over the class of all models) was already observed by Hustadt and Schmidt [1999], although without providing a proof.

We will not claim that it is *desirable* to always remove sort annotations in this way. It could well be the case that these annotations actually help a particular prover to avoid useless inferences, outweighing the additional formula complexity. We simply want to point out that one is actually allowed to remove sort annotations (under certain conditions) in case this provides benefits in practice.

Let us start from the beginning. We first properly formalize what we mean by *sort erasure*.

Definition 6.8. The sort erasure transformation $(\cdot)^-$ takes a many-sorted logic formula to unsorted logic as follows:

$$\begin{aligned} a^- &= a, \text{ for } a \text{ a first-order atom} \\ (\varphi \wedge \psi)^- &= \varphi^- \wedge \psi^- \\ (\neg\varphi)^- &= \neg(\varphi^-) \\ (\exists x{:}\alpha.\varphi)^- &= \exists x.(\varphi)^- \end{aligned}$$

Second, we need to make precise what are the models for the target logic of this sort erasure transformation.

Definition 6.9 (Unsorted hybrid functional models)**.** An unsorted hybrid functional model for a modal signature $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel}\rangle$ is a structure $\langle W, \{f_r \mid r \in \mathsf{Rel}\}, \{de_r \mid r \in \mathsf{Rel}\}, V, g\rangle$ where W is a non-empty set, $V : \mathsf{Prop} \to 2^W$, $g : \mathsf{Nom} \to W$ and, for each $r \in \mathsf{Rel}$, $f_r : W \times W \to W$ and $de_r : 2^W$.

In this case, we say that every unsorted model induces a (relational) hybrid one such that for every relation r the following holds:

$$\forall x, y.r(x, y) \leftrightarrow (\neg de_r(x) \wedge \exists z.f(z, x) = y) \tag{6.9}$$

Clearly, φ is not equivalent to φ^- in the general case. But consider again the Kripke model of Figure 6.1a. It can be represented using three functions (cf. Figures 6.1b and 6.1c); but we can certainly represent it using an additional function i as long as we satisfy this condition:

$$\forall xy.r(x, y) \leftrightarrow (\neg de(x) \wedge (f(x) = y \vee g(x) = y \vee h(x) = y \vee i(x) = y)) \tag{6.10}$$

In general, by stipulating, for example $i(x) = f(x)$ for all x, one can take any structure satisfying condition (6.1) and extend it in order to satisfy also (6.10). This shows that we can pick any $n \geq 3$ and represent the structure of Figure 6.1a using n total functions (and a predicate de). But can we represent it with less than three functions? The answer is clearly "no": since node a has a fan-out of three, we require at least three functions to witness all the successors of a. Since the maximum fan-out (via a relation r) of a relational structure with domain W is $|W|$ we arrive at the following proposition.

Proposition 6.5. *For any $\mathcal{H}(@)$-formula φ, the following are equivalent:*

1. *φ is satisfiable*

2. *$FT^{\mathcal{H}}(\varphi)$ is satisfiable*

3. *$FT^{\mathcal{H}}(\varphi)^{-}$ is satisfiable*

Proof. From the previous discussion, $FT^{\mathcal{H}}(\varphi)$ is satisfiable iff it is satisfiable by a functional model such that its domains W and I (for sorts ω and ι, respectively) both have the same cardinality. But by using any bijection between W and I we define an unsorted model that satisfies $FT^{\mathcal{H}}(\varphi)^{-}$ (this is shown by a trivial induction). □

Because the number of possible functions of $W \rightarrow W$ is $|W|^{|W|}$, this cardinality argument is not compatible with maximal models. However, using classical preservation results it is not hard to see that we can do with a weaker form of maximality, that is, we only need to have all the realizations of functions for worlds that are "reachable" from the initial point of evaluation.

Definition 6.10 ($\overline{w}$-maximal models)**.** Consider an unsorted hybrid function model $\mathcal{I} = \langle W, I, \{f_r \mid r \in \mathsf{Rel}\}, \{de_r \mid \mathsf{Rel}\}, V, g\rangle$, and let $r^{\mathcal{I}}$ be the relation induced by (6.9) for each $r \in \mathsf{Rel}$. Moreover, let $\overline{w}$ be a non-empty subset of W, and let $W_{\overline{w}}$ be the generated submodel (cf. Definition 1.14) of the hybrid model induced by $\mathcal{I}$ whose domain is the smallest one containing $\overline{w}$. We say $\mathcal{I}$ is *$\overline{w}$-maximal* if for each total function $\gamma : W_{\overline{w}} \rightarrow W_{\overline{w}}$ such that $(v, \gamma(v)) \in r^{\mathcal{I}}$ for all $v \in W_{\overline{w}}$, there exists an $i \in W$ for which $f_r(i, x) = \gamma(x)$, for all x.

In order to obtain an equivalent version of Theorem 6.3 we need a syntactic way of distinguishing variables that stand for function indices (former sort ι) from those that stand for domain elements (former sort ω). The first class is composed of those variables x that occur in a term of the form $f_r(x, t)$, while the second class is of those that occur as $f_r(y, x)$. Prefix-stability will guarantee disjointedness of these classes.

Definition 6.11 (Domain variables)**.** The set of *domain variables* of a formula φ in the functional correspondence language is defined as:

$$\{x \mid f_r(y, x) \text{ occurs in } \varphi, \text{ for some variable } y, \text{ and } x \text{ is free}\}$$

Theorem 6.5. *Let φ be a prefix-stable formula in the functional correspondence language, and let y be a free variable in φ that occurs in context $f_r(y, t)$, with all the variables in t free in φ. Moreover, let $z_1 \ldots z_n$ be the domain variables of φ. Finally, let $\mathcal{I}$ be a $w_1 \ldots w_n$*-maximal *functional model. Then, for every valuation v such that $v(z_1) = w_1, \ldots v(z_n) = w_n$, we have:*

$$\mathcal{I} \models_{\text{FO}} \forall x_1 \ldots x_k \exists y.\varphi[v] \text{ iff } \mathcal{I} \models_{\text{FO}} \exists y \forall x_1 \ldots x_k.\varphi[v]$$

The proof is analogous to the one for Theorem 6.3. Now, while Proposition 6.3 shows that in the many-sorted case we can assume a satisfiable formula is satisfied by a maximal model, what can we say about satisfiability by $\overline{w}$-maximal models? For the basic case, the answer is simple.

Proposition 6.6. *An formula φ of $\mathcal{H}(@, \downarrow)$ is satisfiable iff there exists an unsorted hybrid functional model $\mathcal{I}$ such that;*

1. *$\mathcal{I}$ induces a model that satisfies φ at some world w,*
2. *$\mathcal{I}$ is w-maximal.*

Proof. We only need to prove the left-to-right direction, so let the hybrid model $\mathcal{M} = \langle W, R, V, g\rangle$ be such that $\mathcal{M}, w \models \varphi$ and pick any unsorted functional model $\mathcal{I} = \langle W, \{f_r \mid r \in \mathsf{Rel}\}, \{de_r \mid r \in \mathsf{Rel}\}, V, g\rangle$ such that $\mathcal{I}$ induces $\mathcal{M}$. Now, let $\Gamma_r = \{\gamma \mid \neg de_r(v) \text{ implies } (v, \gamma(v)) \in R(r), \forall v\}$ for each $r \in \mathsf{Rel}$, and $\Gamma = \bigcup \Gamma_r$. Define then an unsorted functional model $\mathcal{I}' = \langle W \cup \Gamma, \{f'_r \mid r \in \mathsf{Rel}\}, \{de'_r \mid r \in \mathsf{Rel}\}, V', g\rangle$, where $de'_r = de_r \cup \Gamma$, $V'(p) = V(p)$ for all $p \in \mathsf{Prop}$, and, for every $r \in \mathsf{Rel}$, $f'_r : W \cup \Gamma \times W \cup \Gamma \to W \cup \Gamma$ is any arbitrary function that satisfies $f'_r(\gamma, v) = \gamma(v)$ for all $\gamma \in \Gamma_r$ and all $v \in W$. It is straightforward to verify that $\mathcal{I}'$ is w-maximal. Moreover, the identity on W is a bisimulation between $\mathcal{M}$ and the model induced by $\mathcal{I}'$, so the latter must also satisfy φ at w. $\square$

Using Proposition 6.6 it is simple to reproduce the proof of Theorem 6.4 and obtain the following result.

Theorem 6.6. *Let φ be a formula of $\mathcal{H}(@, \downarrow)$. The following are equivalent:*

1. *φ is satisfiable.*
2. *$\exists x{:}\omega.OFT^{\mathcal{H}}_x(\varphi)$ is satisfiable.*
3. *$\exists x.OFT^{\mathcal{H}}_x(\varphi)^-$ is satisfiable.*

Now, what happens if we are interested in satisfiability with respect to some particular class of models? The first thing to observe is that the proof of Proposition 6.6 does not always go through. For example, it breaks if we are interested in satisfiability with respect to the class of models where relation r satisfy the seriality condition:

$$\forall x \exists y.r(x, y) \tag{6.11}$$

This class is captured by the modal axiom $[r]p \to \langle r\rangle p$. The reason why the proof is not valid is simply that the w-maximal model obtained does not satisfy the frame condition (6.11). What we will see next is that we can give a different, general proof that works whenever we are reasoning with respect to a first-order definable class of models closed by disjoint unions. The class defined by (6.11) falls in this category. In fact, by a well-known result due to Goldblatt and Thomason, every class of models that is both first-order and modally definable (that is, definable by a basic modal formula) must be closed by disjoint unions [Goldblatt and Thomason, 1975]. Hence, this will be a very general result.

We begin by defining an operation Ψ_κ on models. Intuitively, $\Psi_\kappa(\mathcal{M})$ is the model obtained by taking κ isomorphic copies of $\mathcal{M}$.

Definition 6.12. Let $\mathcal{M} = \langle W, R, V, g\rangle$ be a hybrid model over signature $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel}\rangle$ and let κ be an ordinal, then $\Psi_\kappa(\mathcal{M})$ is defined as the model $\langle W', R', V', g'\rangle$, where:

$$\begin{aligned} W' &\overset{def}{=} \kappa \times W, \\ R'(r) &\overset{def}{=} \{((a,w),(a,v)) \mid a \in \kappa \text{ and } (w,v) \in R(r)\}, \\ V'(p) &\overset{def}{=} \kappa \times V(p), \\ g'(i) &\overset{def}{=} (0, g(i)). \end{aligned}$$

Clearly, we have the following bisimulation, for all κ: $\mathcal{M}, w \underline{\leftrightarrow} \Psi_\kappa(\mathcal{M}), (0, w)$. In fact, when restricted to Kripke models (i.e., ignoring the clause for g') this operation is isomorphic to taking disjoint unions of $\mathcal{M}$ with itself κ times. That means that if a class of models $\mathcal{C}$ is modally definable, then $\mathcal{C}$ is closed by Ψ_κ.

Proposition 6.7. *Let $\mathcal{C}$ be a class of models that is closed by Ψ_κ and let φ be formula of $\mathcal{H}(@, \downarrow)$. Then, φ is satisfiable with respect to $\mathcal{C}$ iff there exists an unsorted hybrid functional model $\mathcal{I}$ such that:*

1. *$\mathcal{I}$ induces a model that is in $\mathcal{C}$ and satisfies φ at some world w,*
2. *$\mathcal{I}$ is w-maximal.*

Proof. The argument is very similar to that of Proposition 6.6. Given a hybrid model $\mathcal{M}$ with domain W such that $\mathcal{M}, w \models \varphi$ we first build the model $\mathcal{M}' = \Psi_\kappa(\mathcal{M})$ with $\kappa = |W|^{|W|}$. By construction, $\mathcal{M}'$ is in $\mathcal{C}$ and since it is bisimilar to $\mathcal{M}$ it also satisfies φ at world $(0, w)$. It is easy to turn any functional model inducing $\mathcal{M}'$ into a w-maximal one. $\square$

Corollary 6.1. *Let $\mathcal{C}$ be a class of modally definable models let φ be a formula of $\mathcal{H}(@, \downarrow)$. The following are equivalent:*

1. *φ is satisfiable in $\mathcal{C}$.*
2. *$\exists x{:}\omega.OFT^{\mathcal{H}}_x(\varphi)$ is satisfiable in $\mathcal{C}$.*
3. *$\exists x.OFT^{\mathcal{H}}_x(\varphi)^-$ is satisfiable in $\mathcal{C}$.*

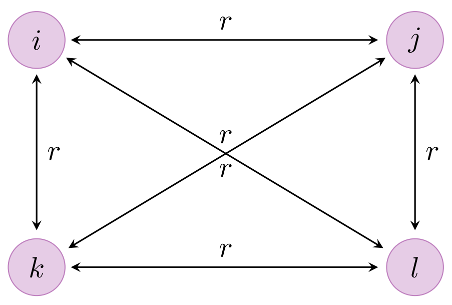

Figure 6.3: A $\mathcal{C}$-model for ψ (relation s is omitted).

This result covers every class of models definable with a basic modal axiom. What about classes definable with hybrid axioms (i.e., formulas containing nominals)? The definable classes in this case may be but are not necessarily closed under Ψ_κ. In fact, we will close this chapter exhibiting a class of models, definable by a hybrid formula, for which sort erasure is not sound in the case of the optimized translation.

Consider the hybrid axiom $\langle s\rangle i$. It is well-known that it modally defines the class of models that satisfy the first-order formula:

$$\forall xy.s(x,y) \tag{6.12}$$

Since, under this class, s behaves like a universal modality it is possible to combine it with the expressive power of hybrid logics to impose cardinality conditions on a model. For example, the following formula is satisfiable only by models with exactly four elements, each of them labeled i, j, k and l, respectively:

$$[s](i \lor j \lor k \lor l) \land @_i\neg j \land @_i\neg k \land @_i\neg l \land @_j\neg k \land @_j\neg l \land @_k\neg l \tag{6.13}$$

Now, let ψ be the conjunction of formula (6.13) with the following additional formulas:

$$@_i(\langle r\rangle j \land \langle r\rangle k \land \langle r\rangle l \land [r]\neg i) \tag{6.14}$$
$$@_j(\langle r\rangle i \land \langle r\rangle k \land \langle r\rangle l \land [r]\neg j) \tag{6.15}$$
$$@_k(\langle r\rangle i \land \langle r\rangle j \land \langle r\rangle l \land [r]\neg k) \tag{6.16}$$
$$@_l(\langle r\rangle i \land \langle r\rangle j \land \langle r\rangle k \land [r]\neg l) \tag{6.17}$$

Clearly, ψ is satisfiable in $\mathcal{C}$. In fact, any model for ψ will be isomorphic (modulo the valuation) to the one in Figure 6.3. Finally, let φ be the conjunction of the following formulas:

$$[s]\langle r\rangle(i \vee j) \tag{6.18}$$
$$[s]\langle r\rangle(i \vee k) \tag{6.19}$$
$$[s]\langle r\rangle(i \vee l) \tag{6.20}$$
$$[s]\langle r\rangle(j \vee k) \tag{6.21}$$
$$[s]\langle r\rangle(j \vee l) \tag{6.22}$$
$$[s]\langle r\rangle(k \vee l) \tag{6.23}$$

The model of Figure 6.3 also satisfies φ and, therefore, satisfies $\psi \wedge \varphi$. Now, consider the formula $OFT^{\mathcal{H}}_{x}(\psi \wedge \varphi)$, which has to be satisfiable in $\mathcal{C}$ as well. What is the minimum number of elements of sort ι that a $\mathcal{C}$-model for $OFT^{\mathcal{H}}_{x}(\psi \wedge \varphi)$ requires? We will see that at least six are necessary (the precise answer is, in fact, "exactly six", but we won't show the upper-bound) and since it also requires exactly four elements of sort ω we will conclude that sorts cannot be safely removed in this case.

To see this, first observe that we have an upper bound for the number of elements of sort ι required, and it is given by the number of existentially quantified variables in $OFT^{\mathcal{H}}_{x}(\psi \wedge \varphi)$, which coincides with the number of diamonds in $\psi \wedge \varphi$; namely, eighteen. We need a function for each of these existentially quantified variables, and the minimum number of elements of sort ι is simply the minimum number of distinct functions needed.

We will restrict our attention to formulas (6.18)–(6.23). Exactly one diamond occurs in each of them, and we will therefore informally say that we require a function to satisfy each of these six formulas. In the end, we will conclude that six distinct functions are needed for this.

Let $\mathcal{M}$ be the model of Figure 6.3 and let $|\mathcal{M}| = \{i, j, k, l\}$. Consider formula (6.18) and assume γ_1 is a function that can be used to satisfy it. Regardless the initial point of evaluation the $[s]$ is universal, so, for every w in $\mathcal{M}$, we must have $\mathcal{M}, w \models \langle r\rangle(i \vee j)$, and, therefore, $\mathcal{M}, \gamma_1(w) \models (i \vee j)$. Since $\mathcal{M}$ is irreflexive, this means γ_1 must satisfy $\gamma_1(i) = j$ and $\gamma_1(j) = i$. We can do this analysis for all the six formulas and find that these are the constraints to be fulfilled:

$$\begin{array}{lll}
\gamma_1(i) = j & \gamma_2(i) = k & \gamma_3(i) = l \\
\gamma_1(j) = i & \gamma_2(j) \in \{i, k\} & \gamma_3(j) \in \{i, l\} \\
\gamma_1(k) \in \{i, j\} & \gamma_2(k) = i & \gamma_3(k) \in \{i, l\} \\
\gamma_1(l) \in \{i, j\} & \gamma_2(l) \in \{i, k\} & \gamma_3(l) = i \\
\\
\gamma_4(i) \in \{j, k\} & \gamma_5(i) \in \{j, l\} & \gamma_6(i) \in \{k, l\} \\
\gamma_4(j) = k & \gamma_5(j) = l & \gamma_6(j) \in \{k, l\} \\
\gamma_4(k) = j & \gamma_5(k) \in \{j, l\} & \gamma_6(k) = l
\end{array}$$

$$\gamma_4(l) \in \{j, k\} \qquad \gamma_5(l) = j \qquad \gamma_6(l) = k$$

Because γ_1, γ_2 and γ_3 differ in the value for i, they must be all distinct functions. Similarly, γ_4 differs from γ_1 in the value for j, from γ_2 in the value for k and from γ_3 everywhere. The same can be said about γ_5 and γ_6 and, therefore, we may conclude that six distinct functions are needed to satisfy these six formulas.

Part III

Direct resolution

Chapter 7

Ordered direct resolution with selection

The direct resolution calculus DR introduced in Chapter 2, although sound and complete, is simply not suitable for a realistic theorem-prover implementation: because *every formula* in each clause may lead to some inference, the set of clauses will tend to grow in an unmanageable way for all but the most trivial cases.

In this chapter we will turn DR into an *ordered resolution* calculus. That is, we will use an ordering on formulas to restrict which formulas in a clause may participate in inferences. Selection functions will be used as a mechanism to optionally override the default ordering-based method of selection of formulas.

We will have to give suitable admissibility conditions on the ordering with which we can prove refutational completeness. Contrasting with the case of resolution in first-order logic where partial orderings are used to account for unification of non ground terms, we will work with total orderings. Hence, at most one formula in each clause will be available for inferences. This property simplifies implementations and may result in efficiency gains.

For the completeness proof we will actually establish the *reduction property for counterexamples* from which it will also follow that the calculus is compatible with the standard redundancy criterion. The reader is assumed to be familiarized with the proof scheme presented in Chapter 2.

This chapter (as well as the following one) is closely based on [Areces and Gorín, 2009]. Some proofs were omitted or reduced to only the most relevant cases. The complete versions can be found in the technical appendix of the paper. Throughout this chapter we will assume all formulas to be in negation normal form.

7.1 The ordered direct resolution calculus $\mathsf{DR}_S^{\succ}$

We begin by formalizing our notion of selection function. Our focus is on selection functions that pick at most one formula per clause. In the case of first-order logic, selection formulas typically may choose only negative literals and we will follow essentially the same approach. However, as we work with clauses which can contain arbitrary @-formulas from $\mathcal{H}(@,\downarrow)$, we will not use the concept of "negative literals" when defining selection functions but rather that of "not being a positive literal". In this case, the set of positive literals PLit is defined as:

$$\mathsf{PLit} ::= @_i j \mid @_i p \mid @_i\langle r\rangle j \tag{7.1}$$

for $i, j \in \mathsf{Nom}$, $p \in \mathsf{Prop}$ and $r \in \mathsf{Rel}$. Observe that in classical resolution being a negative literal is the same as not being a positive literal.

Definition 7.1 (Selection function)**.** A *selection function* S assigns to each clause C a set of @-formulas $S(C)$ such that $S(C) \subseteq C$, $|S(C)| \leq 1$ and $S(C) \cap \mathsf{PLit} = \emptyset$.

Figure 7.1 defines the ordered direct resolution calculus $\mathsf{DR}_S^{\succ}$, parameterized over an ordering on formulas $\succ$ and a selection function S. The leftmost premise in each binary rule is always the *main premise.* We denote as $\mathit{ClSet}^*_{\mathrm{S}\succ}(\varphi)$ the minimum set that contains $\mathit{ClSet}(\varphi)$ and is closed under the rules of $\mathsf{DR}_S^{\succ}$.

Observe that $\mathsf{DR}_S^{\succ}$ differs from DR only in its global and side conditions. The side conditions prevent certain redundant inferences by: i) enforcing a normal form on equalities (rules SYM and PAR); ii) making the choice of the main premise unique (rule PAR) and iii) avoiding useless skolemizations (rule $\langle r\rangle$). The global conditions, on the other hand, ensure that only one formula in each clause may be involved in inferences. We will call this formula the *distinguished formula* of the clause.

Definition 7.2 ($\mathit{max}^{\succ}$ and $\mathit{dist}^{S\succ}$)**.** Given an ordering $\succ$ and a selection function S, we define $\mathit{max}^{\succ}(C)$ as the maximum formula (with respect to $\succ$) in C, and $\mathit{dist}^{S\succ}(C)$ as the function such that $\mathit{dist}^{S\succ}(C) = \varphi$ whenever either $S(C) = \{\varphi\}$, or both $S(C) = \emptyset$ and $\mathit{max}^{\succ}(C) = \varphi$.

We define next a class of orderings for which we will guarantee refutational completeness (Section 7.4) using the notion of Herbrand model introduced in Section 7.3.

7.2 Admissible orderings

In a strict sense, any ordering $\succ$ such that $\mathsf{DR}_S^{\succ}$ is refutationally complete would be *admissible.* This notion of admissibility, undeniably general, would

---

$$\text{RES}\quad \frac{C \vee @_i\neg p \quad D \vee @_i p}{C \vee D} \qquad\qquad \text{REF}\quad \frac{C \vee @_i\neg i}{C}$$

$$\text{SYM}\quad \frac{C \vee @_j i}{C \vee @_i j}\ \dagger \qquad\qquad \text{PAR}\quad \frac{C \vee \varphi(i) \quad D \vee @_i j}{C \vee D \vee \varphi(i/j)}\ \ddagger$$

$$\wedge\quad \frac{C \vee @_i(\varphi_1 \wedge \varphi_2)}{C \vee @_i\varphi_1 \quad C \vee @_i\varphi_2} \qquad\qquad \vee\quad \frac{C \vee @_i(\varphi_1 \vee \varphi_2)}{C \vee @_i\varphi_1 \vee @_i\varphi_2}$$

$$[r]\quad \frac{C \vee @_i[r]\varphi \quad D \vee @_i\langle r\rangle j}{C \vee D \vee @_j\varphi} \qquad\qquad \langle r\rangle\quad \frac{C \vee @_i\langle r\rangle\varphi}{C \vee @_i\langle r\rangle j \quad C \vee @_j\varphi}\ \star$$

$$@\quad \frac{C \vee @_i@_j\varphi}{C \vee @_j\varphi} \qquad\qquad \downarrow\quad \frac{C \vee @_i{\downarrow}j.\varphi(j)}{C \vee @_i\varphi(j/i)}$$

Side conditions

$\dagger$ $i \succ j$

$\ddagger$ $i \succ j$ and $\varphi(i) \succ @_i j$

$\star$ $\varphi \notin \mathsf{Nom}$ and $j \in \mathsf{Nom}$ is *fresh*.

Global conditions

- in $C \vee \psi$, ψ must be selected by S or $\succ$-maximum in the clause.
- in $D \vee \psi$, ψ is $\succ$-maximum in the clause and nothing is selected.

Figure 7.1: The ordered direct resolution calculus $\mathsf{DR}^{\succ}_{S}$, for S a selection function and $\succ$ an ordering.

not be of much use. Instead, we will give the name "admissible" to a non-empty class of orderings satisfying certain conditions that can be effectively checked. We will later prove that any admissible ordering (in this sense) induces a complete calculus (in fact, a calculus with the reduction property for counterexamples).

Throughout this section, we will use some classical notions that come from term-rewriting theory but have found applications in automated theorem proving. We include formal definitions next for the sake of self-containness and refer the reader to any textbook on term-rewriting for more details (e.g., [Baader and Nipkow, 1998]).

Definition 7.3. A binary relation $\succ$ is called an *ordering* if it is transitive and irreflexive; if, additionally, for any two distinct elements x and y one of $x \succ y$ or $y \succ x$ holds, $\succ$ is said to be *total.* An ordering $\succ$ is called *well-founded* when there is no infinite chain $x_1 \succ x_2 \succ x_3 \ldots$

Let $\succ$ be an ordering on formulas, and let's indicate with $\varphi[\psi]_p$ a formula φ where ψ appears at position p. We say that $\succ$ has the *subformula property* if $\varphi[\psi]_p \succ \psi$ whenever $\varphi[\psi]_p \neq \psi$, and that it is a *rewrite ordering* when $\varphi[\psi_1]_p \succ \varphi[\psi_2]_p$ iff $\psi_1 \succ \psi_2$.

A well-founded rewrite ordering is called a *reduction ordering*, and if it also has the subformula property, it is called a *simplification ordering.*

We will typically work with a lifting to clauses $\succ_c$ of an ordering on formulas $\succ$. We require $\succ_c$ to be total, well-founded and to satisfy that if $C \succ_c D$, then either $D = \emptyset$ or $max^{\succ}(C) \succeq max^{\succ}(D)$. We include, for the sake of completeness, a possible lifting.

Definition 7.4. For $\succ$ a total ordering on formulas, $\succ_c$ is the unique ordering on clauses such that $C \succ_c D$ if and only if $C \neq \emptyset$, and either

- $D = \emptyset$, or
- $max^{\succ}(C) \succ max^{\succ}(D)$, or
- $max^{\succ}(C) = max^{\succ}(D)$ and $C \setminus max^{\succ}(C) \succ_c D \setminus max^{\succ}(D)$.

It is easy to see that Definition 7.4 is just a specialization of the multiset ordering to the case of finite sets. Therefore, $\succ_c$ is a total ordering, and well-founded whenever $\succ$ is well-founded too (see [Baader and Nipkow, 1998]).

From now on, we will use $\succ$ to denote both an ordering on formulas an its lifting to clauses. The one thing to keep in mind is that whenever we say that $C_1 \succ C_2$ for clauses $C_1 = @_i\psi_1 \vee @_j\psi_2$ and $C_2 = @_k\chi_1 \vee @_l\chi_2$ this must be understood as $\{@_i\psi_1, @_j\psi_2\} \succ_c \{@_k\chi_1 \vee @_l\chi_2\}$ and not as $@_i\psi_1 \vee @_j\psi_2 \succ @_k\chi_1 \vee @_l\chi_2$ (of course, since a disjunction of @-formulas is not a singleton clause, there is no ambiguity).

Definition 7.5 (Admissible orderings)**.** We will say that an ordering $\succ$ on $\mathcal{H}(@, \downarrow)$-formulas is *admissible for* $\mathsf{DR}_S^{\succ}$ whenever it satisfies the following conditions, for all $\varphi, \psi \in \mathcal{H}(@, \downarrow)$ and all $i, j \in \mathsf{Nom}$:

A1 – $\succ$ is a total simplification ordering,

A2 – $\varphi \succ i$ for all $\varphi \notin \mathsf{Nom}$,

A3 – if $\varphi \succ \psi$, then $@_i\varphi \succ @_j\psi$,

A4 – if ψ is a proper subformula of φ, then $\varphi \succ \psi(i/j)$,

A5 – $[r]i \succ \langle r\rangle j$.

Definition 7.5 looks rather arbitrary but this is because it is simply listing conditions that are used throughout the completeness proof. We shall motivate them by way of examples.

- Conditions A1 and A3 induce a notion of subformula property for @-formulas (e.g., $@_i\langle r\rangle\varphi \succ @_j\varphi$). This is used in the proof of the *Main premise reduction lemma* (i.e., Lemma 7.1).
- Conditions A2 and A3 imply that equalities (i.e., formulas of the form $@_ij$) are the smallest @-formulas, and this will be important when proving the *Upwards and downwards preservation lemmas.*
- Condition A4 is required to guarantee that $@_i{\downarrow}j.\varphi \succ @_i\varphi(j/i)$ in the proof of the *Main premise reduction lemma.* It is also used in the *Upwards preservation lemma.*
- Condition A5 is needed, for example, to guarantee that, in the $[r]$ rule, the side premise is smaller than the main premise and, thus, that the *Main premise reduction lemma* holds.

Now that we have in place our notion of admissible ordering, we are able to give our first step in following the proof-scheme outlined in Section 2.1.

Lemma 7.1 (Main premise reduction for $\mathsf{DR}_S^{\succ}$)**.** *Let $\succ$ be an admissible ordering. If C is the main premise of an inference of $\mathsf{DR}_S^{\succ}$ with a conclusion D, then $C \succ D$.*

Proof. The proof is straightforward but rather tedious due to the numerous rules to consider. We will show the interesting case, namely, that of rule $[r]$. Hence, let $C = C' \vee @_i[r]\varphi$ and let $E = E' \vee @_i\langle r\rangle j$ be the premises of an inference producing $D = C' \vee E' \vee @_j\varphi$. Because of A1, $\succ$ has the subformula property and, thus, $[r]\varphi \succ \varphi$ which implies, by A3, $@_i[r]\varphi \succ @_j\varphi$. Observe also that it must be the case $@_i[r]\varphi \succ @_i\langle r\rangle j$: by A2 there exists some nominal k such that $\varphi \succeq k$ which implies $[r]\varphi \succeq [r]k$ and, because of A5, $[r]\varphi \succeq [r]k \succ \langle r\rangle j$. We conclude that $@_i[r]\varphi \succ @_i\langle r\rangle j \succ \psi$, for all $\psi \in E'$; so, of ψ is a formula in D and $\psi \succ @_i[r]\varphi$, then $\psi \in C'$. Finally, since obviously $@_i[r]\varphi \notin D$ we conclude, from Definition 7.4, that $C \succ D$. $\square$

Finally, we only need to show that the conditions in Definition 7.5 are not too restrictive, and that there actually exist orderings satisfying them. We will exhibit one such ordering, based on the so-called Knuth-Bendix ordering (KBO), after its first use by Knuth and Bendix [1970].

The KBO is an ordering on the set $\text{Term}(\mathcal{F}, \mathcal{V})$ of first-order terms defined over the set of function symbols $\mathcal{F}$ and set of variables $\mathcal{V}$. It is parameterized by a *precedence* $>$, that is a partial ordering on $\mathcal{F}$, and a *weight function* $w : \mathcal{F} \cup \mathcal{V} \to \mathbb{N}$. The weight function must be compatible with respect to $>$, this means: i) $w(x) = \mu$, for all $x \in \mathcal{V}$, for some $\mu > 0$ and ii) if $f \in \mathcal{F}$ is a unary function symbol with $w(f) = 0$, then $f > g$ for all $g \in \mathcal{F}$, $g \neq f$. The weight function w is extended to terms by $w(f(t_1, \ldots t_m)) = w(f) + w(t_1) + \cdots + w(t_n)$.

Definition 7.6 (KBO). Let $>$ be a precedence on $\mathcal{F}$, w a weight function and $s, t \in \text{Term}(\mathcal{F}, \mathcal{V})$. Then $s \succ_{\text{kbo}} t$ iff

- $s = f(s_1, \ldots s_n)$, $t = g(t_1 m \ldots t_m)$, and
  1. $|s|_x \geq |t|_x$, $\forall x \in \mathcal{V}$. $|u|_x$ is "the occurrence count of x in u", and

  2a. $w(s) > w(t)$, or

  2b. $w(s) = w(t)$, $f > g$, or

  2c. $w(s) = w(t)$, $f = g$, and $\exists k. s_1 = t_1, \ldots s_{k-1} = s_{k-1}, s_k \succ_{\text{kbo}} t_k$.
- $s = f(s_1, \ldots s_n)$, $t = x \in \mathcal{V}$ and x occurs in s.

If a precedence $>$ is total, then any KBO extension of $>$ is a total simplification ordering on ground terms (see, e.g. [Baader and Nipkow, 1998]).

We plan to use KBO to define an admissible ordering for $\mathcal{H}(@, \downarrow)$. Therefore, we will consider every $\mathcal{H}(@, \downarrow)$-formula as a ground term over the set of function symbols

$$\mathcal{F} = \mathsf{Prop} \cup \mathsf{Nom} \cup \mathsf{Rel} \cup \{\neg, \wedge, \vee, @, \downarrow, \langle\ \rangle, [\ ]\}$$

with the obvious arities, the only proviso being that the nominal argument of every @-formula is considered as the rightmost argument in the corresponding term (e.g., the formula $@_i\langle r\rangle p$ will correspond to the term $@(\langle\ \rangle(r, p), i)$).

Proposition 7.1. *Let $>$ be any total ordering on O and let $w : O \to \mathbb{N} \setminus \{0\}$ be any weight function such that*

1. $w(i) = w(j)$ *for all* $i, j \in \mathsf{Nom}$
2. $w(f) > w(i)$ *for all* $f \in O \setminus \mathsf{Nom}$, $i \in \mathsf{Nom}$
3. $w([\ ]) > w(\langle\ \rangle)$.

Then, $\succ_{\text{kbo}}$ is an admissible ordering, where $\succ_{\text{kbo}}$ is the Knuth-Bendix ordering based on $>$ and w.

Proof. Since $>$ is total, $\succ_{\text{kbo}}$ is a total simplification ordering on ground terms. Condition 2 guarantees that $w(\varphi) > w(i)$ for all $\varphi \notin \mathsf{Nom}$ and, thus, A2 holds. For A3, observe that, from the Definition 7.6, if $\varphi \succ_{\text{kbo}} \psi$ (and $w(\varphi) = w(\psi)$) then $@(\varphi, i) \succ @(\psi, j)$. If φ has a proper subformula ψ, it is because φ is of the form $f(\varphi_1, \dots \varphi_k)$ and, from Condition 2 we have $w(f) > 0$ which implies $w(\varphi(\psi)) > w(\psi)$. But from Condition 1 we have $w(\psi) = w(\psi(i/j))$ and, thus, $\varphi \succ_{\text{kbo}} \psi(i/j)$, which establishes A4. Finally, A5 follows trivially from Conditions 1 and 3. □

7.3 Herbrand models for hybrid logics

In order to carry out the completeness proof sketched in Chapter 2 we shall provide a suitable notion of Herbrand model for $\mathcal{H}(@, \downarrow)$. But before going into the details of this, we ought to give the abstract conditions we expect such models to satisfy.

There are two features of classical Herbrand models we want to mimic. First, we want Herbrand models to be syntactic in nature: in first-order logic, the domain of a Herbrand model is the set of all ground terms of the language (or a partition of that set if dealing with equality) and, thus, the interpretation function for constants and function symbols is trivial. Second, we want to mirror the fact that any set of ground first-order atoms Γ induces a Herbrand model H_Γ such that $H_\Gamma \models \Gamma$. With this in mind, we are now ready to define hybrid Herbrand models.

Definition 7.7 ($\sim_I$)**.** Given $I \subseteq \mathsf{PLit}$, define $\sim_I \subseteq \mathsf{Nom} \times \mathsf{Nom}$ as the reflexive, symmetric and transitive closure of $\{(i, j) \mid @_i j \in I\}$. $\mathsf{Nom}/_{\sim_I}$ is the set of equivalence classes of $\sim_I$, and $[i]_I$ is the equivalence class assigned to i by $\sim_I$. We will usually write $[i]$ instead of $[i]_I$ when I is clear from context.

Definition 7.8 (Hybrid Herbrand models)**.** A *hybrid Herbrand model* is just a set $I \subseteq \mathsf{PLit}$. Furthermore, let $\langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel} \rangle$ be the signature of PLit and $i \in \mathsf{Nom}$; we will say that $I, i \models \varphi$ iff $\mathcal{M}^I, [i] \models \varphi$, where $\mathcal{M}^I = \langle W^I, R^I, V^I, g^I \rangle$ with

$$\begin{array}{lcl} W^I & = & \mathsf{Nom}/_{\sim_I} \\ R^I(r) & = & \{([i], [j]) \mid @_i \langle r \rangle j \in I\} \\ V^I(p) & = & \{[i] \mid @_i p \in I\} \\ g^I(i) & = & [i]. \end{array}$$

Summing up, we identify hybrid Herbrand models with sets of positive literals, and interpret them as hybrid models whose domain is a partition of the set of all nominals.

Proposition 7.2. *If I is a hybrid Herbrand model, then $I \models I$.*

Proof. Straightforward from Definition 7.8. □

7.4 Refutational completeness of $\mathsf{DR}_S^{\succ}$

We are now ready to prove that if $\succ$ is an admissible ordering, then $\mathsf{DR}_S^{\succ}$ is refutationally complete. In what follows we take $\succ$ to be a fixed admissible ordering (cf. Definition 7.5).

The only ingredient still missing from the sketch in Chapter 2 is the model-building procedure. Before going into its formal definitions, let us explain what we will be trying to achieve.

Candidate models for N are hybrid Herbrand models defined using ε_C, i.e., the contribution of each clause C to the final model. Because we want every productive clause (i.e., clauses with a non-empty contribution) to be a potential side premise for a binary rule, we will stipulate that only clauses C such that $S(C) = \emptyset$ and $max^{\succ}(C) \in \mathsf{PLit}$ may be productive.

Moreover, in order to properly deal with equality, we require an additional technical property on every productive clause C: the contribution of C must not be reducible by paramodulation with another productive clause. A similar requirement is usually demanded in the proof of completeness for other paramodulation-based calculi (cf. Nieuwenhuis and Rubio [2001]). This notion of *reducedness* is properly formalized in Definition 7.11. Observe, however, that it must be necessarily defined along with ε_C in a mutually recursive way. In defining this reduced form, we will use a substitution σ_I of nominals by the smallest nominal in the equivalence class induced by a Herbrand interpretation I.

Definition 7.9 (σ_I)**.** Given a hybrid Herbrand interpretation I, we define the substitution of nominals by nominals:

$$\sigma_I = \{i \mapsto j \mid i \sim_I j \wedge (\forall k)((k \sim_I j \wedge k \neq j) \implies k \succ j)\}.$$

In words, σ_I substitutes each nominal with the least nominal of its class, which is taken as the class representative. We now define the set Simp of formulas that cannot be further simplified using unary rules.

Definition 7.10 (Simp)**.** The set of *simple formulas* of $\mathcal{H}(@,\downarrow)$ is defined as:

$$\mathsf{Simp} ::= @_i j \text{ (with } i \succ j) \mid @_i p \mid @_i \neg a \mid @_i \langle r \rangle j \mid @_i [r] \varphi$$

where $i, j \in \mathsf{Nom}$, $p \in \mathsf{Prop}$, $a \in \mathsf{Atom}$, $r \in \mathsf{Rel}$ and $\varphi \in \mathcal{H}(@,\downarrow)$.

We define next the candidate model building procedure. Observe that Definitions 7.11 and 7.12 are mutually recursive.

Definition 7.11 (Reduced form)**.** Let C be a clause and $\varphi = max^{\succ}(C)$. If $\varphi \in \mathsf{Simp}$ and either a) $\varphi \in \mathsf{PLit}$ and $\varphi = \varphi\sigma_{I_C}$, or b) $\varphi = @_i[r]\psi$ and $i = i\sigma_{I_C}$, then we say that both φ and C are in *reduced form.*

Definition 7.12 (I_N, I^C, I_C and ε_C)**.** Let N be an arbitrary set of clauses, C an arbitrary clause (not necessarily in N), and let $\varphi = max^{\succ}(C)$.

- I_N, a *candidate model* for N, is defined as $I_N = \bigcup_{C \in N} I^C$.
- I^C, the *partial interpretation of* N *above* C is defined as $I^C = I_C \cup \varepsilon_C$.
- I_C, the *partial interpretation of* N *below* C is defined as $I_C = \bigcup_{C \succ D} \varepsilon_D$.
- ε_C, *the contribution of* C *to the candidate model*, is defined as $\varepsilon_C = \{\varphi\}$ whenever it simultaneously holds that:

  1. $C \in N$,
  2. C is in reduced form,
  3. $\varphi \in \mathsf{PLit}$,
  4. $I_C \not\models C$, and
  5. $S(C) = \emptyset$.

  and $\varepsilon_C = \emptyset$ otherwise. If $\varepsilon_C \neq \emptyset$ then we call C *productive.*

Note that, because of the admissibility conditions, equalities are the smallest @-formulas. Hence, if C is productive and $\mathit{dist}^{S\succ}(C)$ is an equality then every formula in C must also be an equality. Furthermore, if $\mathit{dist}^{S\succ}(C)$ is not an equality, then $\sigma_{I_C} = \sigma_{I_D}$ for all $D \succ C$.

From here on when not specified otherwise, N is taken to be an arbitrary but fixed set of clauses, and C an arbitrary but fixed clause not necessarily in N.

Lemma 7.2 (Downwards preservation for $\mathsf{DR}_S^{\succ}$). *If $I_N \not\models C$, then $I_C \not\models C$.*

Proof. We prove the contrapositive form, so assume, for the sake of contradiction, that for some $\varphi \in C$, $I_C \models \varphi$ but $I_N \not\models \varphi$. Observe that it cannot be the case $\varphi \in \mathsf{PLit}$. Now, consider the least $D \succeq C$ such that $I_D \models \varphi$ but $I^D \not\models \varphi$. From Definition 7.12 there are only three cases that we have to consider, namely: $\varepsilon_D = \{@_i j\}$, $\varepsilon_D = \{@_i p\}$ and $\varepsilon_D = \{@_i \langle r \rangle j\}$. We will only look here at the last one. In this case there must exist ψ_1 and ψ_2 such that $[r]\psi_2$ is a subformula of ψ_1 and $\varphi = @_k \psi_1$. But, by conditions A1, A2 and A5 of Definition 7.5, $\psi_1 \succ [r]\psi_2 \succ \langle r \rangle j$, and, thus, we get $@_i \langle r \rangle j \succeq \mathit{max}^{\succ}(C) \succeq \varphi \succ @_i \langle r \rangle j$. □

By requiring productive clauses to be in reduced form, we can give a syntactic description of equalities occurring in I_N that allows us to prove the *Upwards preservation lemma.*

Lemma 7.3. *If $i\sigma_{I_N} \neq i$, then I_N contains only one equality where i occurs, and it is of the form $@_i j$ with $j = j\sigma_{I_N}$.*

Lemma 7.4 (Upwards preservation for $\mathsf{DR}_S^{\succ}$). *Let D be the consequent of an inference with main premise C. If $I_N \not\models C$ and $I_C \not\models D$, then $I_N \not\models D$.*

Proof. Because of the *Main premise reduction lemma*, $max^{\succ}(C) \succeq \varphi$ for all $\varphi \in D$. Since $I_N \not\models C$, we already know $I_N \not\models max^{\succ}(C)$. Hence, we can reduce the proof to showing that, for any φ, if $max^{\succ}(C) \succ \varphi$ and $I_C \not\models \varphi$, then $I_N \not\models \varphi$. Now, suppose, for the sake of contradiction, that $E \succeq C$ is the least clause such that φ is true under I^E but false under I_E. Of the three alternatives, here we will only consider the most interesting one, namely, $\varepsilon_E = \{@_i j\}$.

Clearly, for this to be possible φ must be of the form $@_k l$. Now, by Lemma 7.3, we get that either $\varepsilon_E \subset \{@_k l, @_l k\}$, or $\varepsilon_E \subset \{@_k m, @_l m\} \subset I^E$. However, the latter cannot be true since that would imply $k \succ m$ and $l \succ m$ and, because $\succ$ is a rewrite ordering, we would have $@_k l \succ @_k m$ and $@_k l \succ @_l m$ (notice, for the second case, that if $k \succ l$ then we may conclude that $@_k l \succ @_k m \succ @_l m$, while, if $l \succ k$, then $@_k l \succ @_l k \succ @_l m$). Thus, $\varepsilon_E \subset \{@_k l, @_l k\}$ should hold. However, $\varepsilon_E = \{@_k l\}$ cannot be the case, since that would imply $@_k l \succeq max^{\succ}(C) \succ @_k l$. Finally, if $\varepsilon_E = \{@_l k\}$, then $l \succ k$ and $@_k l \succ @_l k \succeq \varphi \succ @_k l$. □

An inspection of the above proof shows that we can actually assert a more general result: if $max^{\succ}(C) \succ \varphi$ and $I_C \not\models \varphi$ then $I_D \not\models \varphi$ for all $D \succeq C$. From this, we get the following:

Corollary 7.1. *If C is a productive clause and $\varphi \in C$ but $\varepsilon_C \neq \{\varphi\}$, then $I_D \not\models \varphi$ for all $D \succeq C$.*

Lemma 7.5. *Let $C \in N$ be such that $C \neq \emptyset$ and $I_C \not\models C$. If C is not productive, then there exists an inference in $\mathsf{DR}^{\succ}_{S}$ such that*

1. *C is the main premise*
2. *the side premise (if present) is productive, and*
3. *some consequent E is such that $I_C \not\models E$.*

Proof. Let $\varphi = dist^{S\succ}(C)$. If $\varphi \notin \mathsf{Simp}$, C is trivially the premise of some unary rule and the proposition holds. Now, suppose $\varphi \in \mathsf{Simp}$ is not in reduced form; this means, using Lemma 7.3, that some clause D (with $C \succ D$) contributes an $@_i j$ for an i occurring in φ. It is easy to check that, in this case, PAR can be applied on D and C. Finally, if φ is in reduced form, it must be of the form $@_i \neg i$ (note that $@_i \neg j$ cannot be in reduced form if $I_C \models @_i j$ and $i \neq j$), $@_i \neg p$ or $@_i [r]\psi$. We show how to proceed in the last case using the $[r]$ rule; the remaining two cases are analogous.

For $I_C \not\models @_i[r]\psi$ to happen, it must be the case that, for some nominal j, $I_C, i \models \langle r \rangle j$ but $I_C, j \not\models \psi$. This implies, together with the fact that C is in reduced form, that $@_i \langle r \rangle k \in I_C$ for some k such that $I_C \models @_j k$. Therefore, there must exist a clause D such that $C \succ D$ and $\varepsilon_D = \{@_i \langle r \rangle k\}$ which, hence, may be the side premise in an instance of the $[r]$ rule with C as the main premise. Now, let $E = C' \vee D' \vee @_j\psi$, where $C = C' \vee @_i[r]\psi$ and $D = D' \vee @_i\langle r\rangle k$ be the consequent of the inference. $I_C \not\models E$ follows from:

1. $I_C \not\models C$ implies $I_C \not\models C'$,
2. $C \succ D$ implies (using Corollary 7.1) $I_C \not\models D'$, and
3. $I_C \models @_jk$ and $I_C \not\models @_k\psi$, implies $I_C \not\models @_j\psi$.

$\square$

We can finally put all the pieces together as was sketched in Chapter 2. Lemmas 7.2, 7.4 and 7.5 fit together nicely into a *Counterexample lemma* which, together with Lemma 7.1, gives us the completeness result.

Theorem 7.1. $\mathsf{DR}_S^{\succ}$ *has the reduction property for counterexamples and, therefore, is refutationally complete.*

Summing up, then, with Theorem 7.1 we have established refutational completeness of $\mathsf{DR}_S^{\succ}$ and, moreover, the proof was obtained by adapting the standard proof for first-order saturation based methods. In addition, except perhaps for the details in the notion of admissible ordering, the framework obtained seems quite natural and, arguably, susceptible to be used for establishing completeness for other hybrid calculi.

Chapter 8

More effective calculi

In this chapter we will pursue the definition of a terminating direct resolution based calculus for $\mathcal{H}(@)$ and we will introduce two refinements of $\mathsf{DR}_S^{\succ}$. The first one will allow us to limit the paramodulation inferences needed to guarantee completeness. This time, however, the completeness proof will have to be less standard and more involved. Finally, this calculus will be extended with machinery to control the generation of witnesses by rule $\langle r \rangle$. The completeness proof will be essentially the same, but additionally we will be able to prove termination for $\mathcal{H}(@)$ (even without redundancy elimination).

8.1 Paramodulation restricted to labels: $\mathsf{DRL}_S^{\succ}$

From the proof of Lemma 7.5 we can see that refutational completeness of $\mathsf{DR}_S^{\succ}$ is preserved even if paramodulation is restricted to Simp formulas (cf. Definition 7.10). What we will see now is that by adding a simple sound rule to the calculus and using a construction slightly more involved, one can repeat the completeness proof of the previous chapter establishing the stronger result that paramodulation inferences can be further restricted to the following rule:

$$\frac{C \vee @_i\varphi \quad D \vee @_ij}{C \vee D \vee @_j\varphi} \quad i \succ j, \varphi \succ j, @_i\varphi \in \mathsf{Simp}.$$

That is, we need not consider any other nominal but the label i of the distinguished formula $@_i\varphi$ of the main premise and we don't have to replace other occurrences of i inside φ. This by itself is a nice property from a practical point of view. Moreover, by taking advantage of this restriction we will be able to define in Section 8.2 a terminating calculus for $\mathcal{H}(@)$.

In Figure 8.1 we define $\mathsf{DRL}_S^{\succ}$. Observe that not only we replaced PAR by the aforementioned rule, but we added a new inference rule: SYM$^{\neg}$.

$$\text{SYM}^{\neg}\quad \frac{C \vee @_j \neg i}{C \vee @_i \neg j}\ \dagger \qquad\qquad \text{PAR}^{@}\quad \frac{C \vee @_i \varphi \quad D \vee @_i j}{C \vee D \vee @_j \varphi}\ \ddagger$$

Side conditions

† $i \succ j$

‡ $i \succ j$, $\varphi \succ j$ and $@_i\varphi \in \mathsf{Simp}$

Figure 8.1: $\mathsf{DRL}^{\succ}_{S}$ is obtained from $\mathsf{DR}^{\succ}_{S}$ by replacing PAR by SYM$^{\neg}$ and PAR$^{@}$.

The calculus would not be complete without this additional rule. This is witnessed by the following example.

Example 8.1. Consider the set of (singleton) clauses $N = \{C_1, C_2\}$ where $C_1 = @_i j$ and $C_2 = @_j \neg i$ with an ordering such that $i \succ j$. N is evidently unsatisfiable so, if SYM$^{\neg}$ were not required for the completeness of $\mathsf{DRL}^{\succ}_{S}$, one should be able to find a $\mathsf{DRL}^{\succ}_{S}$-derivation of the empty clause that does not use the SYM$^{\neg}$ rule. However, we cannot use PAR$^{@}$ to replace i by j in C_2, and since $i \succ j$ we cannot use SYM on C_1 to derive the singleton $@_j i$ aiming to replace j by i in C_2. Without SYM$^{\neg}$ we would be stuck. But if we use it, we can derive the empty clause as follows:

$$\begin{array}{lll} C_3: & @_i \neg j & \text{(by SYM}^{\neg}\text{ on } C_2) \\ C_4: & @_j \neg j & \text{(by PAR}^{@}\text{ on } C_3 \text{ and } C_1) \\ & \bot & \text{(by REF on } C_4) \end{array}$$

In order to prove the refutational completeness of $\mathsf{DRL}^{\succ}_{S}$ we simply have to "tweak" the constructions used for the completeness proof of $\mathsf{DR}^{\succ}_{S}$ until everything fits together again. It must be said upfront that, in this case, the required notions and definitions are much less intuitive. We will try to motivate them by giving a short account of the problems we will have to face when adapting the proof.

Example 8.2. Let $i \succ j \succ k$ and let the set $N = \{C_1, C_2, C_3\}$, where $C_1 = @_j k$, $C_2 = @_i k \vee @_i j$ and $C_3 = @_i \neg j$. It follows that, for any admissible ordering, $C_3 \succ C_2 \succ C_1$; therefore, we expect C_1 to be productive and, thus, we should have $I_{C_2} = \{@_j k\}$.

If we use the definitions from the completeness proof for $\mathsf{DR}^{\succ}_{S}$, C_2 would have to be non-productive (since it is not in reduced form) and this would make it the minimum counterexample for I_N, only reducible by PAR with

C_1. However, this would not be a valid inference in $\mathsf{DRL}^{\succ}_{S}$. Remember that the notion of "reduced form" was introduced to characterize those clauses that cannot be the main premise of a paramodulation inference using a productive clause as side premise. Hence, we can already see that we need to adjust the notion of "being in reduced form" in order to account for the fact that C_2 cannot be reduced by paramodulation. The natural way to do this would be by demanding only the label of the maximum formula to be reduced (this notion will be called *weak reduced form* in Definition 8.2).

But here comes the tricky part. If C_2 becomes a reduced clause, it will also turn into a productive one. In that case, C_3 would be the minimum counterexample for I_N and the only inference we can draw from it is by using the PAR$^{@}$ rule on C_2 obtaining $D = @_i k \vee @_j \neg j$. However, now $I_N \models D$, and thus we don't obtain a new counterexample. A closer inspection of this example shows that it is actually the *Upwards preservation lemma* that is failing.

Summing up, in order to prove that $\mathsf{DRL}^{\succ}_{S}$ has the reduction property for counterexamples we need an *Upwards preservation lemma.* But from Example 8.2, that implies that the following should hold:

$$I_N \models @_i j \quad I_N \models @_j k \quad I_N \not\models @_i k$$

which is simply not possible (because $\models (@_i j \wedge @_j k) \rightarrow @_j k$). To escape from this apparent dead end, what we do is to drop the $\models$ relation altogether and repeat the steps of the completeness proof we did for $\mathsf{DR}^{\succ}_{S}$, but now in terms of a carefully tailored relation $\models\!\!\approx$ for which we shall have $I_N \models\!\!\approx @_i j$ and $I_N \models\!\!\approx @_j k$ while $I_N \not\models\!\!\approx @_i k$.

Of course, we will also have to ensure that whenever $I_N \models\!\!\approx \varphi \not\models\!\!\approx$, then $I_N \models \varphi$ too, thus from proving that every saturated consistent set of clauses is $\models\!\!\approx$-satisfiable we shall infer that every saturated consistent set of clauses is also $\models$-satisfiable. Before moving to the definition of $\models\!\!\approx$, it is worth observing that a similar example can be devised for relations.

Example 8.3. Let $i \succ j \succ k \succ l$ and let $N = \{C_1, C_2, C_3, C_4\}$ where $C_1 = @_i l$, $C_2 = @_j k$, $C_3 = @_l \langle r \rangle j \vee @_i \langle r \rangle k$, and $C_4 = @_l [r] \neg j$. This time any admissible order entails $C_4 \succ C_3 \succ C_2 \succ C_1$ and, clearly, C_1 and C_2 must be productive clauses. According to the notion of reducedness of Definition 7.11, C_3 would be reducible and, therefore non-productive and the minimum counterexample for I_N. However, no inference can be drawn from C_3. Just like in Example 8.2 everything suggests we need to consider C_3 as a reduced clause, since its distinguished formula is $@_l \langle r \rangle j$ and l is indeed reduced, but this also makes it productive and C_4 becomes the minimum counterexample for I_N. The only clause we can derive from C_4 is $D = @_j \neg j \vee @_i \langle r \rangle k$, but $I_N \models D$ (because $I_N \models @_i \langle r \rangle k$) and therefore we don't obtain a smaller counterexample as required in the completeness proof.

It turns out that Examples 8.2 and 8.3 are sufficiently general, in the sense that every other counterexample can be seen as an instance of one of these two. Essentially, we can say that the exhibited problem relates to some form of *aliasing* due to the presence of nominals: in a productive clause, there is some non-distinguished formula that also becomes true when the distinguished formula is included in the candidate model (e.g. $@_i k$ in C_2 of Example 8.2 and $@_i\langle r\rangle k$ in C_3 of Example 8.3); we shall call them *aliased formulas*.

Fortunately, given some candidate model I_N, we can give a syntactic characterization of all the potential *aliased formulas* even without knowledge of N. So, we will basically stipulate that $I \mathrel{|\!\!\approx} \varphi$ holds whenever $I \models \varphi$ and φ is not potentially aliased, according to I.

Definition 8.1 ($\mathrel{|\!\!\approx}$)**.** We define $\mathrel{|\!\!\approx}$ as the largest relation between hybrid Herbrand models and @-formulas, such that:

1. $I \mathrel{|\!\!\approx} \varphi$ implies $I \models \varphi$,
2. $I \mathrel{|\!\!\approx} @_i j$ iff $I \mathrel{|\!\!\approx} @_j i$,
3. $I \mathrel{|\!\!\approx} @_i j$ and $i \succ j$ implies that for no k such that $I \models @_j k$ and $k \succ j$, $@_i k \in I$,
4. $I \mathrel{|\!\!\approx} @_i\langle r\rangle j$ implies that for no k and l such that $I \models @_i k$, $I \models @_j l$ and $l \succ j$, $@_k\langle r\rangle l \in I$.

Revisiting Example 8.2 may help grasp the ideas behind Definition 8.1.

- $I_N \mathrel{|\!\!\approx} @_j k$ holds because $@_j k \in I_N$ and no other equality labeled with j occurs in I_N.
- $I_N \mathrel{|\!\!\approx} @_i j$ holds for analogous reasons.
- Now, observe that $I_N \mathrel{|\!\!\approx} @_i k$ does not hold, because $I_N \models @_j k$, $@_i j \in I_N$ but $j \succ k$. The last part is crucial; it implies that $@_i j \succ @_i k$ and, thus, it means that $@_i k$ is a potentially aliased formula in the clause that contributed $@_i j$ to I_N (in fact, in Example 8.2 it is aliased).

The slight asymmetry between cases 3 and 4 in Definition 8.1 is due to the fact that, as shown in Example 8.3, in the case of relations the labels of the maximum formula and the aliased formula may differ[1]. The remaining definitions are more natural.

[1] At this point some readers may wonder if Definition 8.1 couldn't be made simpler, e.g., by reducing case 3 to "$I \mathrel{|\!\!\approx} @_i j$ implies $@_i j \in I$". We encourage those readers to verify that with the simpler definition no counterexample can be inferred from the minimum counterexample for I_N, when N is $C_1 = @_j k$, $C_2 = @_i k$, $C_3 = @_k j$, $C_4 = @_i j$ and $C_5 = @_i \neg j$; with $i \succ j \succ k$.

Definition 8.2 (Weak reduced form)**.** Let C be a clause and $\varphi = max^{\succ}(C)$. If $\varphi \in \mathsf{Simp}$ is of the form $@_i\psi$ with $i = i\sigma_{I_C}$, then we say that both φ and C are in *weak reduced form.*

As in the previous section, let N be an arbitrary but fixed set of clauses.

Definition 8.3 (ε_C for $\mathsf{DRL}^{\succ}_{S}$)**.** Let C be a clause (not necessarily in N) and let $\varphi = max^{\succ}(C)$. If it simultaneously holds that:

1. $C \in N$,
2. C is in weak reduced form,
3. $\varphi \in \mathsf{PLit}$,
4. $I_C \not\models_{\approx} C$, and
5. $S(C) = \emptyset$

then $\varepsilon_C = \{\varphi\}$; otherwise, $\varepsilon_C = \emptyset$.

Lemma 8.1 (Main clause reduction for $\mathsf{DRL}^{\succ}_{S}$)**.** *If C is the main premise of an inference of $\mathsf{DRL}^{\succ}_{S}$ and D is one of its conclusions, then $C \succ D$.*

Proof. For SYM$^{\neg}$, the property follows from its side-condition. For the rest of the rules, it follows from the Main clause reduction lemma for $\mathsf{DR}^{\succ}_{S}$. $\square$

Lemma 8.2 (Downwards preservation for $\mathsf{DRL}^{\succ}_{S}$)**.** *If $I_N \not\models_{\approx} C$, then $I_C \not\models_{\approx} C$*

Proof. The proof runs similar to that of the equivalent lemma for $\mathsf{DR}^{\succ}_{S}$. We take $D \succeq C$ to be the least clause such that $I_D \models_{\approx} C$ but $I^D \not\models_{\approx} C$. Now, it must be the case that for some $\varphi \in C$, $I_D \models_{\approx} \varphi$, but this implies that $I_D \models \varphi$. At this point, one can copy almost verbatim the proof of Lemma 7.2 to conclude that it must also be the case that $I^D \models \varphi$. Hence, if $I^D \not\models_{\approx} \varphi$, it must be because one of Conditions 3 or 4 of Definition 8.1 does not hold.

For the first case, assume $\varphi = @_ij$ and, thus, $I^D \not\models_{\approx} @_ij$. Since we know $I_D \models @_ij$, it must be the case that for some $k \succ j$, $I^D \models @_jk$ and $@_ik \in I^D$. This opens up two possibilities:

1. $@_ik \in I_D$. Since $I_D \models @_ij$, $I_D \models @_jk$, but that would imply $I_D \not\models_{\approx} @_ij$.
2. $\varepsilon_D = \{@_ik\}$. We know $I_D \models @_ij$, but since D is productive, it must be in weak reduced form, hence, $j\sigma_{I_D} = i$ which implies $j \succeq i$. But since $@_ik \in \mathsf{Simp}$, $i \succ k$ and, by hypothesis, $k \succ j$, which leads to $j \succeq i \succ k \succ j$.

For the second case, let $\varphi = @_i\langle r\rangle j$. Since $I_D \models_{\approx} @_i\langle r\rangle j$, I_D must contain formulas other than equalities and ϵ_D cannot be an equality. Hence, it must be the case that $\epsilon_D = \{@_l\langle r\rangle k\}$ for l and k such that $l \succ j$, $I_D \models @_ik$ and $I_D \models @_jl$. But since D is in weak reduced form we have $l\sigma_{I_D} = l$ which implies $j\sigma_{I_D} = l$ and, thus, $j \succeq l$. This contradicts $l \succ j$. $\square$

Observe that as a corollary we get that if C is productive, then $I_N \not\approx C$, which given conditions 3 and 4 of Definition 8.1, was not obvious.

Lemma 8.3 (Upwards preservation for $\mathsf{DRL}_S^{\succ}$)**.** *Let D be the consequent of an inference whose main premise is C. If $I_N \not\approx C$ and $I_C \not\approx D$, then $I_N \not\approx D$.*

Proof. The proof follows that of Lemma 7.4. Thus, let φ be such that $max^{\succ}(C) \succ \varphi$ and $I_C \not\approx \varphi$, and let $E \succeq C$ be the least clause such that $I_E \not\approx \varphi$ but $I^E \approx \varphi$. Of the three possible cases, $\varepsilon_E = \{@_i p\}$ is handled exactly like in the proof for Lemma 7.4 (details can be found in [Areces and Gorín, 2009]). We sketch the procedure for the case where ε_E is an equality, the remaining case runs similarly.

If $\varepsilon_E = \{@_a b\}$ with $a, b \in \mathsf{Nom}$, then $a \succ b$, $a\sigma_{I_E} = a$ and φ has to be an equality. That is, for some $i, j \in \mathsf{Nom}$, $i \succ j$, then either $\varphi = @_i j$ or $\varphi = @_j i$. In any case, we have $\varphi \succeq @_i j$ and also $I \approx @_i j$ iff $I \approx \varphi$. We can arrive to a contradiction proving, by case analysis, that it cannot be the case $a \succ i$ nor $i \succ a$ nor $a = i$. The first two rely only on properties of admissible orders, so we will only cover the last case here.

Let us assume that ε_E is $\{@_i b\}$. If $j \succeq b$, then $@_i j \succeq @_i b = max^{\succ}(E) \succeq max^{\succ}(C) \succ \varphi \succeq @_i j$. Now suppose $b \succ j$. For this case, we will rely on Condition 3 of Definition 8.1. Since $I^E \approx @_i j$, it must be the case $I^E \models @_i j$. Now, from this and $@_i b \in I^E$, we get $I^E \models @_j b$, but since $b \succ j$, $@_i b \succ @_i j$ and, thus, we get the contradiction $I^E \not\approx @_i j$ $\square$

The path is finally downhill: from here we can essentially repeat all the steps that lead us to Theorem 7.1. The only additional step is to verify that if the distinguished formula of the minimum counterexample is of the form $@_i \neg j$ and is in weak reduced form, then we must have $j \succ i$ and, thus, the SYM$^{\neg}$ rule is applicable, which is straightforward (details in [Areces and Gorín, 2009]).

Theorem 8.1. $\mathsf{DRL}_S^{\succ}$ *has the reduction property for counterexamples and, therefore, is refutationally complete.*

8.2 On the termination of $\mathsf{DRL}_S^{\succ}$ for $\mathcal{H}(@)$-formulas

As was observed in Chapter 1, $\mathcal{H}(@)$ is a decidable fragment of $\mathcal{H}(@, \downarrow)$. A natural question to ask is then: can we obtain a decision procedure for $\mathcal{H}(@)$ out of $\mathsf{DR}_S^{\succ}$ and/or $\mathsf{DRL}_S^{\succ}$?

There are roughly two ways in which one resolution is shown to be a decision procedure for some class of formulas. The first one is showing that for some orderings, every finite satisfiable set of clauses can be saturated in a finite number of steps. The second one is similar, but one has to show that resolution in conjunction with the eager application of some redundancy elimination techniques (cf. Chapter 2) saturates a finite satisfiable set in finite steps. An example of such a redundancy elimination technique is *condensing*: the replacement of a clause by the smallest instance of the clause which subsumes it.

If any of these condition holds, implementing an effective algorithm that computes this set in finite time is straightforward (e.g., using the "given clause algorithm" [Voronkov, 2001]). Joyner, Jr. [1976] was the first one to study first-order logic resolution as a decision procedure for fragments of the language. Schmidt [1999] shows that resolution with condensing decide the optimized functional translation of various classic modal logics (cf. Chapter 6).

We will say that a calculus that saturates every satisfiable set of clauses in finite steps is *terminating*. So, let's consider the question if there are admissible orderings for which $\mathsf{DR}^{\succ}_{S}$ and/or $\mathsf{DRL}^{\succ}_{S}$ are terminating. We will assume that rule $\langle r \rangle$ is applied only once per clause. The previous completeness proofs justify this assumption. Unfortunately, the answer is strongly negative.

Proposition 8.1. *There is a formula φ such that for every admissible ordering $\succ$ and every selection function S,* $\mathsf{DR}^{\succ}_{S}$ *and* $\mathsf{DRL}^{\succ}_{S}$ *require infinite steps to saturate ClSet(φ).*

Proof. Consider the formula $@_i[r](i \vee @_i\langle r \rangle p) \wedge @_i\langle r \rangle p$. Figure 8.2a show an infinite derivation starting from this formula. In this derivation only rules $\wedge$, $\vee$, $@$, $\langle r \rangle$ and $[r]$ are used and, therefore, is a valid derivation both for $\mathsf{DR}^{\succ}_{S}$ and $\mathsf{DRL}^{\succ}_{S}$. Observe that every non-singleton clause contains exactly one formula that is not an equality. By admissibility of $\succ$ this formula must be the maximum one in the clause and since equalities are not selectable, this cannot be overridden by any selection function. $\square$

What Proposition 8.1 is saying is that to find a decision procedure for $\mathcal{H}(@)$ based on direct resolution we need either a redundancy elimination strategy or a more refined calculus. In any case, observe that as no other symbols but nominals are introduced by the calculus, and given that formulas in consequents are never larger (in number of operators) than those in the antecedent, if we can control the generation of nominals we will ensure termination[2]. So let's begin by analyzing the conditions that give rise to the generation of infinite nominals.

There are essentially two ways in which an infinite number of nominals can be introduced by the rules of $\mathsf{DR}^{\succ}_{S}$ (or $\mathsf{DRL}^{\succ}_{S}$) when applied to a $\mathcal{H}(@)$-formula:

Type 1. A formula of the form $@_i\langle r \rangle\varphi$ introduces a new nominal which, in turn, contributes to the derivation of a new clause containing $@_i\langle r \rangle\varphi$.

[2]When devising terminating resolution strategies for fragments of first-order logic, the related problem one usually has to face is *term-depth* growth.

$$
\begin{array}{rll}
C_1\colon & @_i[r](i \vee @_i\langle r\rangle p) \wedge @_i\langle r\rangle p & \\
C_2\colon & @_i[r](i \vee @_i\langle r\rangle p) & (\text{by } \wedge \text{ on } C_1) \\
C_3\colon & @_i\langle r\rangle p & (\text{ditto}) \\
C_4\colon & @_i\langle r\rangle k_1 & (\text{by } \langle r\rangle \text{ on } C_3) \\
C_5\colon & @_{k_1}(i \vee @_i\langle r\rangle p) & (\text{by } [r] \text{ on } C_2 \text{ and } C_4) \\
C_6\colon & @_{k_1}i \vee \underline{@_{k_1}@_i\langle r\rangle p} & (\text{by } \vee \text{ on } C_5) \\
C_7\colon & @_{k_1}i \vee \underline{@_i\langle r\rangle p} & (\text{by } @ \text{ on } C_6) \\
C_8\colon & @_{k_1}i \vee \underline{@_i\langle r\rangle k_2} & (\text{by } \langle r\rangle \text{ on } C_7) \\
C_9\colon & @_{k_1}i \vee \underline{@_{k_2}(i \vee @_i\langle r\rangle p)} & (\text{by } [r] \text{ on } C_2 \text{ and } C_8) \\
C_{10}\colon & @_{k_1}i \vee @_{k_2}i \vee \underline{@_{k_2}@_i\langle r\rangle p} & (\text{by } \vee \text{ on } C_9) \\
C_{11}\colon & @_{k_1}i \vee @_{k_2}i \vee \underline{@_i\langle r\rangle p} & (\text{ditto}) \\
 & \vdots & \\
C_{15}\colon & @_{k_1}i \vee @_{k_2}i \vee @_{k_3}i \vee \underline{@_i\langle r\rangle p} & (\text{by } \vee \text{ on } C_{14}) \\
 & \vdots & \\
C_{4n+3}\colon & @_{k_1}i \vee \cdots \vee @_{k_n}i \vee \underline{@_i\langle r\rangle p} & (\text{by } \vee \text{ on } C_{4n+2})
\end{array}
$$

(a) Type 1: $@_j\langle r\rangle p$ introduces k_n which is used to produce $@_j\langle r\rangle p$ again. The maximum formula of each non-singleton clause is underlined.

$$
\begin{array}{rll}
C_1\colon & @_i[r](i \wedge \langle r\rangle p) \wedge @_i\langle r\rangle p & \\
C_2\colon & @_i[r](i \wedge \langle r\rangle p) & (\text{by } \wedge \text{ on } C_1) \\
C_3\colon & @_i\langle r\rangle p & (\text{ditto}) \\
C_4\colon & @_i\langle r\rangle k_1 & (\text{by } \langle r\rangle \text{ on } C_3) \\
C_5\colon & @_{k_1}(i \wedge \langle r\rangle p) & (\text{by } [r] \text{ on } C_2 \text{ and } C_4) \\
C_6\colon & @_{k_1}i & (\text{by } \wedge \text{ on } C_5) \\
C_7\colon & @_{k_1}\langle r\rangle p & (\text{ditto}) \\
C_8\colon & @_{k_1}\langle r\rangle k_2 & (\text{by } \langle r\rangle \text{ on } C_7) \\
C_9\colon & @_i\langle r\rangle k_2 & (\text{by PAR}^{@} \text{ on } C_8 \text{ and } C_6) \\
C_{10}\colon & @_{k_2}(i \wedge \langle r\rangle p) & (\text{by } [r] \text{ on } C_2 \text{ and } C_9) \\
C_{11}\colon & @_{k_2}i & (\text{by } \wedge \text{ on } C_{10}) \\
C_{12}\colon & @_{k_2}\langle r\rangle p & (\text{ditto}) \\
C_{13}\colon & @_{k_2}\langle r\rangle k_3 & (\text{by } \langle r\rangle \text{ on } C_{12}) \\
C_{14}\colon & @_i\langle r\rangle k_3 & (\text{by PAR}^{@} \text{ on } C_{13} \text{ and } C_{11}) \\
 & \vdots & \\
C_{19}\colon & @_i\langle r\rangle k_4 & (\text{by PAR}^{@} \text{ on } C_{18} \text{ and } C_{16}) \\
 & \vdots & \\
C_{5n-1}\colon & @_i\langle r\rangle k_n & (\text{by PAR}^{@} \text{ on } C_{5n-2} \text{ and } C_{5n-3})
\end{array}
$$

(b) Type 2: Nominals $k_1, k_2, \ldots$ are such that each k_{n+1} is introduced by $@_{k_n}\langle r\rangle p$. i is assumed to be the smallest nominal (wrt. $\succ$).

Figure 8.2: Infinite derivations assuming that $\succ$ is an admissible ordering.

$$\begin{tikzcd}
 & i_1 \\
 & i_2 \\
 & i_3 \\
 & i_4 \\
i_0 \arrow[ruuuu] \arrow[ruuu] \arrow[ruu] \arrow[ru] & \vdots
\end{tikzcd}$$

(a) Type 1 derivations

$$\begin{tikzcd}
i_0 \arrow[r] & i_1 \arrow[r] & i_2 \arrow[r] & i_3 \arrow[r] & \cdots
\end{tikzcd}$$

(b) Type 2 deriations

Figure 8.3: $i \to j$ means j was introduced from $C \vee @_i\langle r\rangle\varphi$ by rule $\langle r\rangle$.

All of these new nominals are immediate successors of i and they are actually representing the same state in the model, but the calculus cannot detect it. This form of infinite derivation was the one used in the proof of Proposition 8.1, illustrated in Figure 8.2a.

Type 2. There is a formula φ and an infinite sequence of distinct nominals $i_0, i_1, i_2, \ldots$ such that, for all $n \geq 0$, some $@_{i_n}\langle r\rangle\varphi$ in the saturated set introduces, by way of rule $\langle r\rangle$, the nominal i_{n+1}. The calculus is exploring a cycle in the model and cannot detect when to stop the search. Figure 8.2b shows an example of this type of derivations.

Figure 8.3 can be used to verify that derivations of Type 1 and 2 cover all the possible cases. If an infinite number of nominals is generated we need to have either *infinite branching* (corresponding to Type 1 derivations) or an *infinite chain* (Type 2 derivations).

This analysis shows that, to ensure termination, we need to impose some control both on the nominals generated by rule $\langle r\rangle$ and on the way chains of nominal successors are treated.

We devote the rest of this chapter to this issue. It turns out that to tame Type 1 derivations it suffices to make the fresh nominal introduced by rule $\langle r\rangle$ on $C \vee @_i\langle r\rangle\varphi$ a function of $@_i\langle r\rangle\varphi$. Controlling Type 2 derivations will require a careful refinement of rule PAR$^@$. The upshot is we will introduce a refinement of $\mathsf{DRL}_S^{\succ}$ for which we will be able to guarantee termination for $\mathcal{H}(@)$ without requiring additional redundancy elimination rules.

Before introducing this final calculus, we will have to take a detour and introduce a hybrid version of Hilbert's ϵ-operator. Terms built using this new operator will replace "fresh" nominals and will provide us the notion of *derivation history*) we need to control the generation of infinite chains.

8.3 Hilbert's ϵ-operator and $\mathsf{DRL}\epsilon_S^{\succ}$

The ϵ-operator was introduced by Hilbert as part of his program to establish the consistency of arithmetic by finitary means. A first-order ϵ-term is of the form $\epsilon x.\varphi(x)$ where $\varphi(x)$ is a formula and its intended meaning is: "some element e such that $\varphi(e)$ holds, or an arbitrary element if no such e exists" [Hilbert and Bernays, 1939, Leisenring, 1969]. The ϵ-terms were later investigated in the context of linguistics, philosophy and non-classical logics. From the point of view of automated reasoning, ϵ-terms can be seen as an alternative to skolem functions [Giese and Ahrendt, 1999]. Our interest in ϵ-terms lies in that, unlike "fresh" nominals (i.e., skolem constants) an ϵ-term keeps track of its own derivation history.

In first-order logic enriched with the ϵ-operator, the notion of *formula* and *term* become mutually recursive. In hybrid logics, on the other hand, the boundary between formula and term vanishes; hence, they are an interesting setting on their own in which to introduce and investigate ϵ-operators.

We will enrich $\mathcal{H}(@,\downarrow)$ with ϵ-terms of the form $\epsilon\langle l, r, \varphi\rangle$ (for l a nominal or an ϵ-term and $r \in \mathsf{Rel}$) denoting "an r-successor of l where φ holds if such exists, or any element (not necessarily a successor of l) otherwise". Observe that unlike their first-order cousins, hybrid ϵ-terms do not bind variables.

Definition 8.4 (Syntax of $\mathcal{H}(@,\downarrow,\epsilon)$)**.** We will define the set of ϵ-formulas and the hybrid language $\mathcal{H}(@,\downarrow,\epsilon)$ in a mutually recursive way. Assume a fixed signature $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel}\rangle$. We define the set of ϵ-formulas as $\epsilon\text{-}\mathsf{forms} = \{\epsilon\langle l, r, \varphi\rangle \mid \varphi \in \mathcal{H}(@,\downarrow,\epsilon)\}$. For convenience we will use the set of *labels* $\mathsf{Lab} = \mathsf{Nom} \cup \epsilon\text{-}\mathsf{forms}$ when we don't want to distinguish nominals from ϵ-formulas. Elements of Lab will be denoted $l, m, n, \ldots$ Finally, $\mathcal{H}(@,\downarrow,\epsilon)$ is defined as:

$$\varphi ::= p \mid l \mid \neg\varphi \mid \varphi \wedge \varphi \mid [r]\varphi \mid @_l\varphi \mid \downarrow i.\varphi$$

Observe that ϵ-formulas can occur nested, as in the following formula:

$$@_{\epsilon\langle\epsilon\langle i,r,\neg p\rangle,s,p\rangle}(q \wedge [s]\epsilon\langle i, r, q\rangle) \tag{8.1}$$

The subsets of $\mathcal{H}(@,\downarrow)$ that were used until now are lifted to $\mathcal{H}(@,\downarrow,\epsilon)$ in a trivial way (e.g. we assume $\mathsf{PLit} ::= @_l m \mid @_l p \mid @_l\langle r\rangle m$).

Definition 8.5 (Semantics of $\mathcal{H}(@,\downarrow,\epsilon)$)**.** We shall call *pre-structure* to any tuple $\langle \mathcal{M}, A\rangle$ where $\mathcal{M} = \langle W, R, V, g\rangle$ is a conventional hybrid model, and $A : \epsilon\text{-}\mathsf{forms} \to W$. The satisfaction relation $\models$ is defined as follows:

$$\begin{aligned}
\langle \mathcal{M}, A\rangle, w \models p &\text{ iff } w \in V(p)\\
\langle \mathcal{M}, A\rangle, w \models i &\text{ iff } w = g(i)\\
\langle \mathcal{M}, A\rangle, w \models \epsilon\langle l, r, \varphi\rangle &\text{ iff } w = A(\epsilon\langle l, r, \varphi\rangle)\\
\langle \mathcal{M}, A\rangle, w \models \neg\varphi &\text{ iff } \langle \mathcal{M}, A\rangle, w \not\models \varphi\\
\langle \mathcal{M}, A\rangle, w \models \varphi_1 \wedge \varphi_2 &\text{ iff } \langle \mathcal{M}, A\rangle, w \models \varphi_1 \text{ and } \langle \mathcal{M}, A\rangle, w \models \varphi_2\\
\langle \mathcal{M}, A\rangle, w \models [r]\varphi &\text{ iff } (w, v) \text{ implies } \langle \mathcal{M}, A\rangle, v \models \varphi, \text{ for all } v \in W\\
\langle \mathcal{M}, A\rangle, w \models @_i\varphi &\text{ iff } \langle \mathcal{M}, A\rangle, g(i) \models \varphi
\end{aligned}$$

$$\langle \mathcal{M}, A\rangle, w \models @_{\epsilon\langle l,r,\varphi\rangle}\varphi \text{ iff } \langle \mathcal{M}, A\rangle, A(\epsilon\langle l,r,\varphi\rangle) \models \varphi$$
$$\langle \mathcal{M}, A\rangle, w \models {\downarrow} i.\varphi \text{iff} \langle \mathcal{M}_i^w, A\rangle, w \models \varphi$$

A pre-structure $\langle \mathcal{M}, A\rangle$ will be called a model when the following condition holds: if $\langle \mathcal{M}, A\rangle \models @_l\langle r\rangle\varphi$ then $\langle \mathcal{M}, A\rangle \models @_l\langle r\rangle\epsilon\langle l,r,\varphi\rangle$ and $\langle \mathcal{M}, A\rangle \models @_{\epsilon\langle l,r,\varphi\rangle}\varphi$. In addition, whenever $\langle \mathcal{M}, A\rangle \models @_l m$ implies $A(\epsilon\langle l,r,\varphi\rangle) = A(\epsilon\langle m,r,\varphi\rangle)$ we will say that $\langle \mathcal{M}, A\rangle$ is *closed under renaming of labels.*

Intuitively, a *model* is a pre-structure where A interprets ϵ-formulas correctly. A model closed under renaming of labels is, in a way, minimizing the number of witnesses for ϵ-formulas.

Proposition 8.2. *Let φ be a formula where no ϵ-formula occurs. Then φ is satisfiable iff it is satisfiable by a model closed under renaming of labels.*

Proof. The right-to-left implication is trivial. Now, for the other direction, since no ϵ-formula occurs in φ, that means that $\langle \mathcal{M}, A\rangle \models \varphi$ implies $\mathcal{M} \models \varphi$. So we only need to check that given $\mathcal{M} = \langle W, R, V, g\rangle$ we can always pick an A' such that $\langle \mathcal{M}, A'\rangle$ is closed under renaming of labels. For this, take any total and well-founded ordering on W, let $w_\perp$ be some fixed element of W and simply define:

$$A'(\epsilon\langle l,r,\varphi\rangle) \overset{def}{=} \begin{cases} \min\{w \mid (v(l), w) \in R(m), \mathcal{M}, w \models \varphi\} & \text{if } \mathcal{M} \models @_l\langle r\rangle\varphi \\ w_\perp & \text{otherwise} \end{cases}$$
$$v(l) \overset{def}{=} \begin{cases} g(l) & \text{if } l \in \mathsf{Nom} \\ A'(l) & \text{if } l \in \epsilon\text{-}\mathsf{forms} \end{cases}$$

A' is well-defined and it is trivial to verify that $\langle \mathcal{M}, A'\rangle$ is closed under renaming of labels. □

As we have been doing so far, for the rest of this section we will assume formulas are in negation normal form.

In Figure 8.4 we introduce the calculus $\mathsf{DRL}\epsilon_S^{\succ}$, which we will prove complete for $\mathcal{H}(@,\downarrow)$ and terminating for the fragment $\mathcal{H}(@)$. It is important to keep in mind that $\mathsf{DRL}\epsilon_S^{\succ}$ is intended to be *sound but not complete for* $\mathcal{H}(@,\downarrow,\epsilon)$.

This calculus differs from $\mathsf{DRL}_S^{\succ}$ in only a few aspects: i) the rules are written in terms of formulas that may be labeled by nominals or ϵ-formula), ii) rule $\langle r\rangle_\epsilon$ uses ϵ-formulas instead of fresh nominals and, iii) the PAR$^@$ rule is replaced by four rules: PAR$^@_{\not\Diamond}$, PAR$^@_{\Diamond i}$, PAR$^@_{\Diamond\epsilon}$ and PAR$^{@\downarrow}_{\Diamond\epsilon}$. The first two are simply the PAR$^@$ rule over a restricted domain, namely that where the distinguished formula is not of the form $@_l\langle r\rangle\epsilon\langle m,s,\varphi\rangle$. The last two handle this case. Observe PAR$^@_{\Diamond\epsilon}$ handles the case where $l = m$ and $r = s$. As we will later see, this rule is crucial to avoid the generation of

$$\text{RES} \quad \frac{C \vee @_l\neg p \quad D \vee @_l p}{C \vee D} \qquad\qquad \text{REF} \quad \frac{C \vee @_l\neg l}{C}$$

$$\text{SYM} \quad \frac{C \vee @_m l}{C \vee @_l m} \ \dagger \qquad\qquad \text{SYM}^{\neg} \quad \frac{C \vee @_m\neg l}{C \vee @_l\neg m} \ \dagger$$

$$\text{PAR}^{@}_{\Diamond\epsilon} \quad \frac{C \vee @_l\langle r\rangle\epsilon\langle l,r,\varphi\rangle \quad D \vee @_l m}{\begin{array}{l} C \vee D \vee @_m\langle r\rangle\epsilon\langle m,r,\varphi\rangle \\ C \vee D \vee @_{\epsilon\langle l,r,\varphi\rangle}\epsilon\langle m,r,\varphi\rangle \end{array}} \ \dagger \qquad\qquad \text{PAR}^{@}_{\Diamond i} \quad \frac{C \vee @_l\langle r\rangle i \quad D \vee @_l m}{C \vee D \vee @_m\langle r\rangle i} \ \dagger$$

$$\text{PAR}^{@\downarrow}_{\Diamond\epsilon} \quad \frac{C \vee @_l\langle r\rangle\epsilon\langle n,s,\varphi\rangle \quad D \vee @_l m}{C \vee D \vee @_m\langle r\rangle\epsilon\langle n,s,\varphi\rangle} \ \ddagger \qquad\qquad \text{PAR}^{@}_{\not\Diamond} \quad \frac{C \vee @_l\varphi \quad D \vee @_l m}{C \vee D \vee @_m\varphi} \ \star$$

$$\wedge \quad \frac{C \vee @_l(\varphi_1 \wedge \varphi_2)}{C \vee @_l\varphi_1 \quad C \vee @_l\varphi_2} \qquad\qquad \vee \quad \frac{C \vee @_l(\varphi_1 \vee \varphi_2)}{C \vee @_l\varphi_1 \vee @_l\varphi_2}$$

$$[r] \quad \frac{C \vee @_l[r]\varphi \quad D \vee @_l\langle r\rangle m}{C \vee D \vee @_m\varphi} \qquad\qquad \langle r\rangle_\epsilon \quad \frac{C \vee @_l\langle r\rangle\varphi}{\begin{array}{l} C \vee @_l\langle r\rangle\epsilon\langle l,r,\varphi\rangle \\ C \vee @_{\epsilon\langle l,r,\varphi\rangle}\varphi \end{array}} \ *$$

$$@ \quad \frac{C \vee @_l@_m\varphi}{C \vee @_m\varphi} \qquad\qquad \downarrow \quad \frac{C \vee @_l{\downarrow}i.\varphi}{C \vee @_l\varphi(i/l)}$$

Side conditions

$\dagger$ $l \succ m$

$\ddagger$ $l \succ m$ and, $l \neq n$ or $r \neq s$

$\star$ $l \succ m$, $\varphi \succ m$, $@_l\varphi \in \mathsf{Simp}$ and $\varphi \neq \langle r\rangle n$

$*$ $\varphi \notin \mathsf{Lab}$

Global conditions

- in $C \vee \psi$, ψ must be selected by S or $\succ$-maximum in the clause.
- in $D \vee \psi$, ψ is $\succ$-maximum in the clause and nothing is selected.

Figure 8.4: The ordered resolution calculus $\mathsf{DRL}\epsilon^{\succ}_{S}$, for S a selection function and $\succ$ an ordering.

infinite chains of nominals (cf. Figure 8.3b). We will also see that the case handled by rule $\mathrm{PAR}^{@\downarrow}_{\Diamond\epsilon}$ (namely, $l \neq m$ or $r \neq s$) can only occur if the input formula contains the $\downarrow$ operator and therefore the name of the rule.

Because of the ϵ-formulas, soundness of the calculus is not self-evident as in the previous cases.

Theorem 8.2. $\mathsf{DRL}\epsilon_S^{\succ}$ *is sound with respect to* $\mathcal{H}(@, \downarrow\epsilon)$ *(and, therefore, to* $\mathcal{H}(@, \downarrow)$*).*

Proof. It is easy to see that, except for $\mathrm{PAR}^{@}_{\Diamond\epsilon}$, all the rules of $\mathsf{DRL}\epsilon_S^{\succ}$ are truth-preserving; that is, every model that makes true the antecedent of a rule satisfies also the consequents. In particular this is true of the $\langle r\rangle_\epsilon$, unlike rule $\langle r\rangle$ in the previous calculi where the fresh nominal had to be conveniently interpreted and was, therefore, satisfaction-preserving. Rule $\mathrm{PAR}^{@}_{\Diamond\epsilon}$, on the other hand, is truth-preserving but only when restricted to the class of models closed under renaming of variables and therefore the whole calculus is sound with respect to this class. But from Proposition 8.2, a $\mathcal{H}(@, \downarrow, \epsilon)$-formula is satisfiable iff it is satisfiable in the class of models closed under renaming of variables, from which soundness trivially follows. $\square$

We now move to the issue of refutational completeness. For this, we first need to extend the notion of admissible order to the new language.

Definition 8.6 (Admissible ordering)**.** We say an ordering $\succ$ on formulas of $\mathcal{H}(@, \downarrow, \epsilon)$ is *admissible* for $\mathsf{DRL}\epsilon_S^{\succ}$ if it satisfies the following conditions, for all $i \in \mathsf{Nom}$, every $l, m \in \mathsf{Lab}$ and all $\varphi, \psi \in \mathcal{H}(@, \downarrow, \epsilon)$:

A1) $\succ$ is a total simplification order

A2) $\varphi \succ l$ for all $\varphi \notin \mathsf{Lab}$

A3) if $\varphi \succ \psi$, then $@_l\varphi \succ @_m\psi$

A4) if ψ is a proper subformula of φ, then $\varphi \succ \psi(l/m)$

A5) $[r]l \succ \langle r\rangle m$

A6) $l \succ m$ implies $\epsilon\langle l, r, \varphi\rangle \succ \epsilon\langle m, r, \varphi\rangle$.

The reader should check that conditions AA1) to AA5) are essentially those of Definition 7.5 but generalized to Lab. It is straightforward to extend the Knuth-Bendix ordering we used in Proposition 7.1 to obtain an ordering that also satisfies Condition AA6). Notice that this condition guarantees that the main premise of rule $\mathrm{PAR}^{@}_{\Diamond\epsilon}$ is greater than its consequents. With this, it is easy to see that the *Main premise reduction lemma* holds.

Theorem 8.3. $\mathsf{DRL}\epsilon_S^{\succ}$ *is refutationally complete for* $\mathcal{H}(@, \downarrow)$*.*

Proof. The proof is an almost verbatim reproduction of the completeness proof we already did for $\mathsf{DRL}^{\succ}_{S}$. The only thing to be adjusted is that now a set I of PLit formulas shall denote the pre-structure $\langle I, A_I\rangle$, where $A_I(l) = [l]$, i.e., the equivalence class of l in I. Observe that for the case where the least counterexample is not in weak-reduced form, we can derive a new counterexample by using one of rules $\mathrm{PAR}^{@}_{\not\diamond}$, $\mathrm{PAR}^{@}_{\diamond\epsilon}$, $\mathrm{PAR}^{@\downarrow}_{\diamond\epsilon}$ or $\mathrm{PAR}^{@}_{\diamond i}$. □

Notice that the calculus $\mathsf{DRL}\epsilon^{\succ}_{S}$ is *not* complete for $\mathcal{H}(@,\downarrow,\epsilon)$. For example, the formula $@_i(\langle r\rangle p \wedge [r]\neg q) \wedge @_{\epsilon\langle i,r,p\vee q\rangle}\neg p$ leads to a saturated set of clauses that does not contain the empty clause, although it is unsatisfiable. This is because if $\langle \mathcal{M}, A\rangle \models @_i(\langle r\rangle p \wedge [r]\neg q)$ then $\langle \mathcal{M}, A\rangle \models @_i\langle r\rangle\epsilon\langle i,r,p\rangle \wedge @_{\epsilon\langle i,r,p\rangle}(p\wedge\neg q)$ and, therefore, $\langle \mathcal{M}, A\rangle \models @_i\langle r\rangle\epsilon\langle i,r,p\vee q\rangle \wedge @_{\epsilon\langle i,r,p\vee q\rangle}(p\wedge\neg q)$. The catch in the proof of Theorem 8.3 is that the pre-structure $\langle I, A_I\rangle$ does not necessarily satisfy the conditions imposed in Definition 8.5 for it to be a model of $\mathcal{H}(@,\downarrow,\epsilon)$.

8.4 $\mathsf{DRL}\epsilon^{\succ}_{S}$ is terminating for $\mathcal{H}(@)$

We finally turn to the problem of proving that there exist admissible orderings $\succ$ such that, when the input formula is in $\mathcal{H}(@)$, $\mathsf{DRL}\epsilon^{\succ}_{S}$ doesn't generate infinite saturated sets. To do this, we define the function $\mathit{level}\colon \mathsf{Lab} \to \mathbb{N}$:

$$\begin{aligned} \mathit{level}(i) &= 0 \\ \mathit{level}(\epsilon\langle l,r,\varphi\rangle) &= \mathit{level}(l)+1. \end{aligned}$$

Now we can define the class of terminating orders.

Definition 8.7. We say $\succ$ is *terminating* if it is admissible for $\mathsf{DRL}\epsilon^{\succ}_{S}$ and for every $l,m \in \mathsf{Lab}$, $\mathit{level}(l) > \mathit{level}(m)$ implies $l \succ m$.

Now we can formally state what we will prove, namely that if $\succ$ is terminating then, for every $\varphi \in \mathcal{H}(@)$, the following conditions hold:

T1(φ): $\{l \mid \mathit{level}(l) = k \text{ and } l \text{ occurs in } \mathit{ClSet}^*_{\mathrm{S}\succ T}(\varphi)\}$ is finite, for all $k \geq 0$.

T2(φ): $\{\mathit{level}(l) \mid l \text{ occurs in } \mathit{ClSet}^*_{\mathrm{S}\succ T}(\varphi)\}$ is finite.

where $\mathit{ClSet}^*_{\mathrm{S}\succ T}(\varphi)$ is the least set that contains $\mathit{ClSet}(\varphi)$ and is closed under the rules of $\mathsf{DRL}\epsilon^{\succ}_{S}$. The reader should check that these conditions guarantee, respectively, that the problems of Type 1 and 2 previously discussed cannot occur. In what follows, $\succ$ is taken to be an arbitrary terminating order.

Theorem 8.4. *For every $\varphi \in \mathcal{H}(@)$, T1($\varphi$) holds.*

Proof. First, two rather trivial observations:

1. If $\epsilon\langle l,r,\psi\rangle$ occurs in $\mathit{ClSet}^*_{\mathrm{S}\succ T}(\varphi)$, then so does l.

2. If $\epsilon\langle l, r, \psi\rangle$ occurs in $ClSet^*_{S\succ T}(\varphi)$, then $\langle r\rangle\psi$ is a subformula of φ.

Clearly the last condition does not hold if the $\downarrow$ operator occurs in φ[3]. Now, let us define $C^k = \{l \mid level(l) = k$ and l occurs in $ClSet^*_{S\succ T}(\varphi)\}$, and proceed by induction on k. C^0 is finite: it contains only the finitely many nominals in $ClSet(\varphi)$. For the inductive case, let us suppose, that C^k is finite but C^{k+1} is not. By Observation 1, it must be the case that for some $l \in C^k$ there exist infinitely many $\langle r_0\rangle\psi_0, \langle r_1\rangle\psi_1, \langle r_2\rangle\psi_2 \ldots$ such that $\epsilon\langle l, r_i, \psi_i\rangle \in C^{k+1}$ for $i \geq 0$. However, this clearly contradicts Observation 2, for φ has only finitely many subformulas. $\square$

To prove that condition T2(φ) holds for $\varphi \in \mathcal{H}(@)$, we need to find an upper bound for the level of the labels that may appear in $ClSet^*_{S\succ T}(\varphi)$. The modal depth $md(\varphi)$ (cf. Definition 1.16) gives us the desired bound. The proof relies heavily on the fact that, as long as the $\downarrow$ operator does not occur in the input formula, terms of the form $\epsilon\langle l, r, \psi\rangle$ may occur only in restricted positions. The following lemma formalizes this statement.

Lemma 8.4. *For all $\varphi \in \mathcal{H}(@)$, $\epsilon\langle l, r, \psi\rangle$ occurs in $ClSet^*_{S\succ T}(\varphi)$ only in the following kind of formulas:*

1. *$@_m\epsilon\langle l, r, \psi\rangle$, with $m \neq \epsilon\langle l, r, \psi\rangle$*
2. *$@_l\langle r\rangle\epsilon\langle l, r, \psi\rangle$*
3. *$@_{\epsilon\langle l,r,\psi\rangle}\theta$, and if an ϵ-term occurs in θ, then $@_{\epsilon\langle l,r,\psi\rangle}\theta$ is also a formula of kind 1 or 2.*

Proof. The proof is by induction on the derivation of a formula where $\epsilon\langle l, r, \psi\rangle$ occurs. For the base case, an ϵ-formula simply cannot occur in φ. For the inductive case, consider the last rule used to derive a formula containing $\epsilon\langle l, r, \psi\rangle$. We discuss only the few interesting cases. It cannot be the case that the SYM$^{\neg}$ rule generates $@_m\neg\epsilon\langle l, r, \psi\rangle$ because the premise would have to be $@_{\epsilon\langle l,r,\psi\rangle}\neg m$ and, since by inductive hypothesis m could not be an ϵ-term, $level(\epsilon\langle l, r, \psi\rangle) > level(m)$ and this implies, by Definition 8.7, that $\epsilon\langle l, r, \psi\rangle \succ m$ which contradicts the side-condition of the rule. For the PAR$^{@}_{\not\approx}$ rule, the interesting case is when both premises are equalities but in that case the side condition guarantees that $@_m m$ cannot be derived. Finally, observe that, by inductive hypothesis, there are no suitable premises for rule PAR$^{@\downarrow}_{\diamond\epsilon}$. $\square$

Now, Lemma 8.4 is roughly saying[4] that, when restricted to input formulas in $\mathcal{H}(@)$, $\epsilon\langle l, r, \psi\rangle$ may occur in $ClSet^*_{S\succ T}(\varphi)$ only as label of @-formulas, or as the right-hand-side of equalities and relations. However,

[3]For an example of this, consider $\varphi = @_i\langle r\rangle{\downarrow}j.\langle r\rangle(j \wedge p)$.

[4] This lemma also formally proves that rule PAR$^{@\downarrow}_{\diamond\epsilon}$ is only required when the input formula contains the $\downarrow$ operator.

in the last case they are restricted to this form: $@_l\langle r\rangle\epsilon\langle l, r, \psi\rangle$ and we know that $level(\epsilon\langle l, r, \psi\rangle) = level(l) + 1$. Hence, in order to show that T2(φ) holds, we only need to find a bound for the level of ϵ-formulas occurring in labels and in the right-hand-side of equalities.

Theorem 8.5. *If $\varphi \in \mathcal{H}(@)$ then:*

- *$@_l\psi$ occurs in $ClSet^*_{S\succ T}(\varphi)$ implies $level(l) + md(\psi) \leq md(\varphi)$*
- *$@_l m$ occurs in $ClSet^*_{S\succ T}(\varphi)$ implies $level(m) \leq md(\varphi)$.*

The proof is carried out by a longish yet straightforward induction on the derivation of formulas. It is presented in full detail in [Areces and Gorín, 2009].

Corollary 8.1. *For all $\varphi \in ClSet^*_{S\succ T}(\varphi)$, T2($\varphi$) holds.*

Since, for $\varphi \in \mathcal{H}(@)$, every formula in $ClSet^*_{S\succ T}(\varphi)$ is made of subformulas of φ and ϵ-terms, and from Theorem 8.4 and Corollary 8.1 there only finitely many of these, it follows that $ClSet^*_{S\succ T}(\varphi)$ must be finite.

Theorem 8.6. $\mathsf{DRL}\epsilon^{\succ}_{S}$ *is a decision procedure for the satisfiability of $\mathcal{H}(@)$.*

Chapter 9

Notes from an implementor

In the last two chapters we developed the theoretical basis that allows an effective implementation of the direct resolution calculus. As Voronkov [2001] points out, a good theory alone is not enough to implement a realistic prover. One needs to combine it with both efficient algorithms and data structures, and clever heuristics.

Fortunately, since all our completeness proofs follow the framework developed by Bachmair and Ganzinger [2001] (cf. Chapter 2), we can take advantage of many crucial techniques developed for resolution-based first-order theorem proving. This is, roughly, the idea that guided us in the design of the prototypic theorem prover HyLoRes.

The aim of this prover was to serve as a proof-of-concept, with the focus set on simplifying its extension with new rules, strategies and language constructs rather than speed. Nevertheless, most, if not all, of the basic simplification and redundancy elimination techniques that one expects to find in a modern saturation-based theorem prover were incorporated.

A large part of the complexity of first-order resolution-based theorem provers comes from the fact that inferences, simplification and redundancy detection need to be done modulo some form of unification. In fact, a lot of research effort has been put in *term indexing* techniques, roughly "techniques for the design and implementation of structures that facilitate rapid retrieval of set of candidate terms satisfying some property (such as generalizations, instances, unifiability, etc.) from a large collection of terms." [Sekar et al., 2001].

It is widely accepted that these term-indexing techniques share a large portion of the merit for the outstanding performance of modern first-order provers. However, this also means that some enhancements that are well-known in theory cannot be implemented until their compatibility with existing term indexing techniques is shown or radically new techniques are developed (Voronkov [2001] cites the *basic strategy* [Nieuwenhuis and Rubio, 1992] as a classical example of this).

The direct resolution calculus, on the other hand, is a ground calculus so there is no need for this kind of complex apparatus. It is, arguably, simpler to implement. Still, we have found that some well-known techniques must be adapted to this setting with a lot of care or they may compromise the completeness of the calculus. Many of these issues were identified only after very long hours of going through derivations that failed to arrive at a contradiction. It is in the best interests of anyone willing to implement the direct resolution calculus that we will present here our findings.

9.1 The architecture of HyLoRes

In HyLoRes we have implemented the calculus $\mathsf{DRL}\epsilon^{\succ}_{S}$ presented in Chapter 8 extended with some rules to handle additional language constructs (cf. Section 9.4). This makes HyLoRes an effective decision procedure for $\mathcal{H}(@)$.

Conceptually, HyLoRes follows the approach described in Chapter 2: it attempts to derive a contradiction while building a saturation up to redundancy of the initial set of clauses. The redundancy criterion used is comprehended by the standard criterion: it include the subsumption principle, the identification of trivially tautological clauses and the observation that the premise of an inference using any of the unary rules of $\mathsf{DRL}\epsilon^{\succ}_{S}$ is always deemed redundant by its consequents.

Like almost every modern saturation-based automated theorem prover, HyLoRes implements a variant of the *given clause algorithm*, whose first use, according to Lusk [1992], dates back to the resolution-based *Functionless Theorem Prover* implemented by Overbeek in the mid seventies. It is regarded today as a versatile algorithm to compute the closure of any set with respect to a collection of inference rules.

In this algorithm, the clause set is partitioned in two: Active and Passive clauses. The important invariant is that every possible inference between Active clauses has already been performed. Passive is a fair priority queue (by "fair" we mean that it is guaranteed that no clause may be queued indefinitely long). At every step, the "most promising" clause (i.e., the one with highest priority) is selected from Passive and every possible inference between this given clause and Active clauses is computed; the given clause then becomes an Active clause. This conceptually simple idea is complicated slightly by the introduction of redundancy elimination checks at some stages.

Figure 9.1 describes the actual given clause loop used in HyLoRes. Notice that newly inferred clauses are temporarily accumulated in a set New prior to entering Passive. Observe also that the main loop may terminate in steps 1 or 3; in the first case, this means the input was unsatisfiable, in the second one, that the clause set is saturated with respect to redundancy

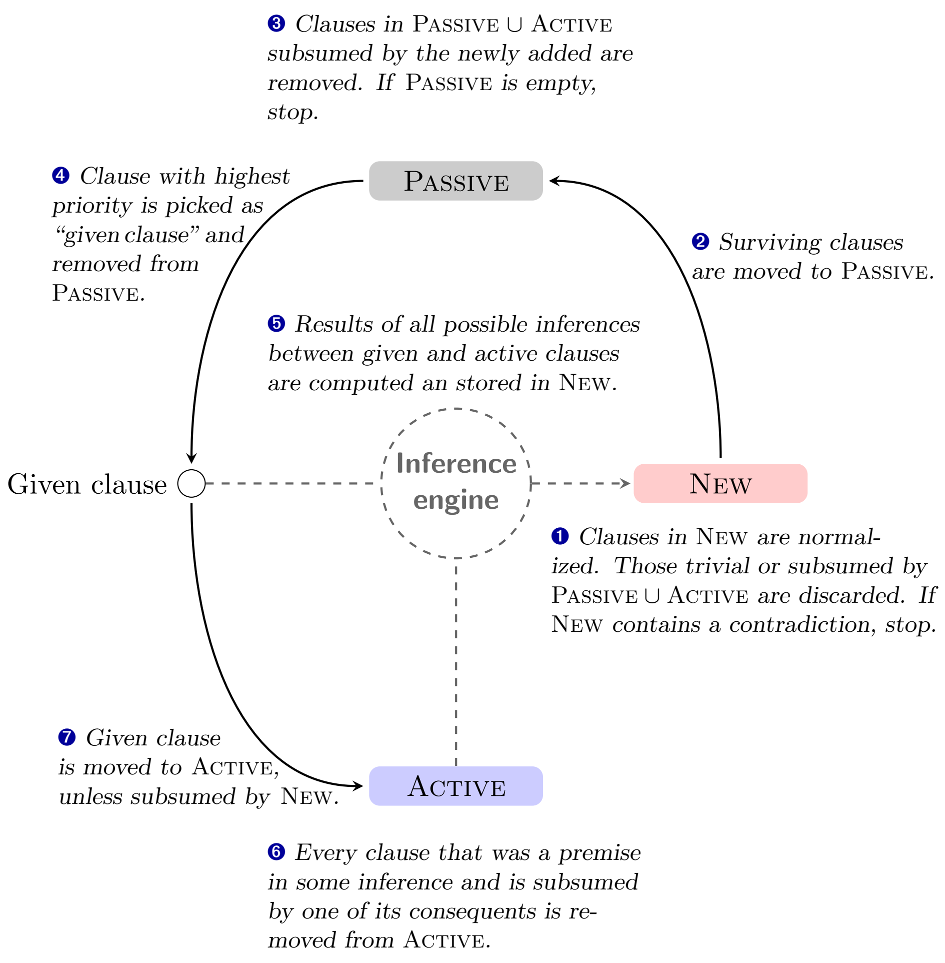
❸ Clauses in PASSIVE ∪ ACTIVE subsumed by the newly added are removed. If PASSIVE is empty, stop.
❹ Clause with highest priority is picked as "given clause" and removed from PASSIVE.
PASSIVE
❷ Surviving clauses are moved to PASSIVE.
❺ Results of all possible inferences between given and active clauses are computed an stored in NEW.
Given clause
Inference engine
NEW
❶ Clauses in NEW are normalized. Those trivial or subsumed by PASSIVE ∪ ACTIVE are discarded. If NEW contains a contradiction, stop.
❼ Given clause is moved to ACTIVE, unless subsumed by NEW.
ACTIVE
❻ Every clause that was a premise in some inference and is subsumed by one of its consequents is removed from ACTIVE.

Figure 9.1: Description of the given clause algorithm used in HyLoRes.

and no contradiction was found (therefore, the input was satisfiable).

There are three distinct subsumption checks, at steps 1, 3, and 6. The first one is usually called a *forward subsumption check* and is known to be crucial in resolution-based theorem proving. The last two are *backward subsumption checks*, where newly derived clauses are checked to see if they subsume a clause already in the clause set. Backward subsumption does not play a role as important as forward subsumption; the check on step 3 may be computationally expensive and, in many cases, it is better to disable it altogether (this can be configured in HyLoRes). On the other hand, the check in step 6 is local, can be performed cheaply on every inference and is therefore always enabled.

Forward subsumption check is a computationally expensive operation.

It is efficiently implemented in HyLoRes using an auxiliary data structure, which represents clauses in Passive $\cup$ Active as an ordered trie: every branch of the trie represents a clause and formulas on each branch occur in ascending order. The forward subsumption check then reduces to finding a branch in the trie that only contain formulas occurring in the clause to be checked.

If a clause in New is found not to be subsumed by the clause set, it is added to the trie and, therefore may be used to subsume the remaining clauses in New. This is why in HyLoRes clauses with fewer formulas are tested for subsumption first.

If the loop ends because Passive is found to be empty (step 3), then Active contains a set of clauses saturated up to redundancy and we can use the candidate model construction to build a model that satisfies all the clauses in Active (as well as the initial formulas). HyLoRes actually implements this feature, which has proved to be invaluable in detecting errors in the implementation: these errors typically compromise completeness which means that the prover will answer "satisfiable" on an unsatisfiable formula. By combining model-generation with model-checking, one can automatically detect bogus satisfiability claims.

9.2 Normalization and tautology elimination

The first step in the given clause algorithm displayed in Figure 9.1 includes the *normalization* of clauses in New. An important aspect of normalization is the *orientation* of equalities and inequalities. That is, for any two labels $l \succ m$, neither $@_m l$ nor $@_m \neg l$ can occur in a normalized clause. Orientation of (in)equalities can be seen as an eager application of the unordered version of rules SYM and SYM$^\neg$ of $\mathsf{DRL}\epsilon_S^{\succ}$ and is therefore a sound operation and is easily shown to preserve completeness. It also means that these rules need not be implemented and that the side conditions of the form $l \succ m$ in the paramodulation rules need not be checked.

Clearly, if a disjunct on some clause is a contradiction then it is safe to remove it from the clause. As part of the clause normalization process, some trivial contradictions (e.g., formulas of the form $@_i\bot$, $@_i\langle r\rangle\bot$ or $@_i\neg i$) are searched for and removed when found. Observe that by removing such contradiction one can obtain an empty clause, proving the unsatisfiability of the input. Because of this form of normalization, it is not necessary to implement rule REF.

Normalization in HyLoRes also involves the pre-calculation of data that is relevant for the internal representation of clauses that enter Passive, but we won't go into those details here.

Tautological clauses are *intuitively* redundant, but they are also "redundant" in the technical sense defined by Bachmair and Ganzinger [2001]. In

$$
\begin{array}{lll}
C_1: & @_j\neg l & \\
C_2: & @_k\langle r\rangle i & \\
C_3: & @_i[r]\neg k & \\
C_4: & @_i k \vee \underline{@_i j} & \\
C_5: & @_j l \vee \underline{@_j k} & \\
C_6: & @_i k \vee \underline{@_j[r]\neg k} & \text{(by } \mathrm{PAR}^{@}_{\not\approx} \text{ on } C_3 \text{ and } C_4\text{)} \\
C_7: & @_j l \vee \underline{@_k\neg l} & \text{(by } \mathrm{PAR}^{@}_{\not\approx} \text{ on } C_1 \text{ and } C_5\text{)} \\
C_8: & @_j l \vee @_i k \vee \underline{@_k[r]\neg k} & \text{(by } \mathrm{PAR}^{@}_{\not\approx} \text{ on } C_6 \text{ and } C_5\text{)} \\
\cancel{C_9}: & @_j l \vee @_i k \vee \underline{@_i\neg k} & \text{(by } [r] \text{ on } C_8 \text{ and } C_2\text{, tautology)}
\end{array}
$$

Figure 9.2: Clauses C_1–C_5 are unsatisfiable, but this derivation fails to arrive at the empty clause. We assume i, j, k, l to be nominals such that $i \succ j \succ k \succ l$. The maximum disjunct of a clause is underlined.

fact, they are trivially comprehended by the standard redundancy criterion. Because we have shown the direct resolution calculi of the previous chapters to be compatible with standard redundancy, it is safe to discard these type of clauses in their implementations without compromising completeness. Or is it?

While experimenting with an early version of HyLoRes we stumbled upon an unsatisfiable formula which the prover could not refute. This, as anyone who implemented a theorem prover knows, is, unfortunately, a fairly typical scenario and simply signals a bug in the implementation. What was rather perplexing in this particular case was that, even after radically reducing the size of the failed derivation, we were unable to identify a missing inference that would explain this failure.

In Figure 9.2 we reproduce this puzzling derivation. The first thing to notice is that clauses C_1–C_5 cannot be simultaneously satisfied. To see this, observe that from C_1 and C_5 a model for these clauses would have to satisfy $j = k$ and, because of C_4, it would additionally have to satisfy $i = j = k$. But C_2 and C_3 cannot be simultaneously true with $i = k$. The second thing to observe is that the derived clause C_9 is a tautology; in fact, one that HyLoRes was able to detect and discard. But once this clause is ignored there is no other inference to be made.

This was a very frustrating example; indeed one which would cast a shadow of doubt on the validity of our completeness proof for $\mathsf{DRL}\epsilon^{\succ}_{S}$. Were the proof correct, then $\mathsf{DRL}\epsilon^{\succ}_{S}$ would possess the reduction property for counterexamples and this example seemed to show this was not the case. It was quite reassuring when we ultimately found out that it was not the completeness of the calculus what we got wrong, but the appropriate notion of tautology, instead.

To better see this, let N be the set containing clauses C_1–C_8. According

C_1:	$@_j \neg l$	
C_2:	$@_k \langle r \rangle i$	
C_3:	$@_i [r] \neg k$	
C_4:	$@_i k \vee \underline{@_i j}$	
C_5:	$@_j l \vee \underline{@_j k}$	
C_6:	$@_i k \vee \underline{@_j [r] \neg k}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_3 and C_4)
C_7:	$@_j l \vee \underline{@_k \neg l}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_1 and C_5)
C_8:	$@_j l \vee @_i k \vee \underline{@_k [r] \neg k}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_6 and C_5)
C_9:	$@_j l \vee @_i k \vee \underline{@_i \neg k}$	(by $[r]$ on C_8 and C_2)
C_{10}:	$@_j l \vee @_i k \vee \underline{@_j \neg k}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_9 and C_4)
C_{11}:	$@_j l \vee @_i k \vee \underline{@_k \neg k}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_{10} and C_5)
C_{12}:	$@_j l \vee \underline{@_i k}$	(by REF on C_{11})
C_{13}:	$@_j l \vee \underline{@_k [r] \neg k}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_3 and C_{12})
$\cancel{C_{14}}$:	$@_j l \vee @_i k \vee \underline{@_k j}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_4 and C_{12}, subs. by C_{12})
$\cancel{C_{15}}$:	$@_j l \vee @_i k \vee \underline{@_k \neg k}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_9 and C_{12}, subs. by C_{12})
C_{16}:	$@_j l \vee \underline{@_i \neg k}$	(by $[r]$ on C_{13} and C_2)
$\cancel{C_{17}}$:	$@_j l \vee @_i k \vee \underline{@_j \neg k}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_{16} and C_4, subs. by C_{12})
C_{18}:	$@_j l \vee \underline{@_k \neg k}$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_{16} and C_{12})
C_{19}:	$@_j l$	(by REF on C_{18})
C_{20}:	$@_l \neg l$	(by $\mathrm{PAR}^{@}_{\not\Diamond}$ on C_3 and C_{19})
C_{21}:	$\bot$	(by REF on C_{20})

Figure 9.3: If C_9 in Figure 9.2 is kept, the empty clause is eventually derived.

to Definition 8.3, $I_N = \{@_j k, @_i j, @_k \langle r \rangle i\}$, which means C_8 is the smallest counterexample, and C_9 is the only inference that can be drawn from it (using a productive clause, C_2, as predicted). By Theorem 8.1 we should have that C_9 must be a counterexample for I_N too, but C_9 is a tautology, how could that be?

Technically speaking, what Theorem 8.1 predicts is $I_N \not\mathrel{|}\!\approx C_9$ and this is indeed the case! To see this, observe that while $I_N \models @_i k$ holds, we have $I_N \not\mathrel{|}\!\approx @_i k$, and this is because $I_N \models @_k j$, $j \succ k$ and $@_i j \in I_N$.

Summing up, one has to keep in mind that both $\mathsf{DRL}^{\succ}_{S}$ and $\mathsf{DRL}\epsilon^{\succ}_{S}$ have the reduction property for counterexamples, but with respect to $\mathrel{|}\!\approx$. This means that the correct notion of standard redundancy for these calculi would be that a clause C is redundant with respect to N if there exist clauses $C_1, \ldots C_k$ in N such that $C_1, \ldots, C_k \mathrel{|}\!\approx C$ and $C \succ C_i$ for all $1 \leq i \leq k$. A clause like C_9 is a tautology with respect to $\models$, but not with respect to $\mathrel{|}\!\approx$ and, therefore, it is not redundant and cannot be eliminated. Figure 9.3 shows that if C_9 is kept, a contradiction is derived.

Similarly, it is not possible to consider $@_i\langle r\rangle j \vee @_i[r]\neg j$ a redundant clause. However, for every Herbrand model I, it is always the case that either $I \models_{\approx} @_i p$ or $I \models_{\approx} @_i\neg p$, so it is safe to remove as redundant a clause of the form $@_i p \vee @_i\neg p$.

9.3 Paramodulation and rewriting

We have seen that $\mathsf{DRL}^{\succ}_S$ (and, therefore, also $\mathsf{DRL}\epsilon^{\succ}_S$) differs from $\mathsf{DR}^{\succ}_S$ in that paramodulation rules only replace labels of distinguished formulas. This is a good property from the point of view of implementation. For example, if the distinguished formula of the given clause is of the form $@_i j$, then we only need to retrieve from Active those clauses whose distinguished formula has i as label and by simply indexing clauses in Active by label this can be done efficiently.

Notice, however, that once a clause is retrieved one may opt to replace every occurrence of the matched nominal in the formula and not only its label. This would be effectively equivalent to adding the PAR rule from $\mathsf{DR}^{\succ}_S$ to the calculus.

In HyLoRes we have implemented paramodulation in such a way that if the distinguished formula of the main premise is of the form $@_i[r]\varphi$, then i is also replaced by j inside φ. Similarly, for $@_i\langle r\rangle i$ we generate $@_j\langle r\rangle j$[1].

As is customary in paramodulation-based reasoning, unit clauses of the form $@_i j$ are handled in a special way in HyLoRes (this was not reflected in Figure 9.1). We will refer to this kind of clauses as *unit equalities.* A clause set that contains a unit equality of the form $@_i j$ can only be satisfied by models where i and j denote the same element and, therefore, it is sound to rewrite *every formula* occurring in *every clause* in the clause set, eliminating i. This transformation can be seen as a generalization of the *unit propagation* rule,[2] and we shall refer to it as *unit rewriting.*

While conceptually simple, a correct implementation of unit rewriting is, in general, tricky. We review next some subtle details that should be kept in mind if implementing $\mathsf{DRL}\epsilon^{\succ}_S$.

In HyLoRes we have decided to rewrite clauses occurring both in Active and Passive (some provers rewrite only active clauses and, from then on, attempt to rewrite every given clause). Therefore, whenever a given clause is a unit equality of the form $@_i j$, every clause occurring in Passive $\cup$ Active that contains nominal i is removed from its set, rewritten, and re-inserted in New. The given clause is then added to Active. A record of occurrences of nominals across clauses must be kept in some data

[1]The reader might wonder if this can interfere with $\models_{\approx}$ in some way, like it was shown in Section 9.2. In this case it is harmless, though: it is easy to see that if I_N is such that $@_i j \in I_N$ and $I_N \not\models_{\approx} @_i\langle r\rangle i$, then, because $i \succ j$, it cannot be the case that $I_N \models_{\approx} @_j\langle r\rangle j$.

[2]Unit propagation, on the other hand, has not been implemented in HyLoRes yet.

structure in order to retrieve the clauses to be rewritten efficiently.

Notice that rewriting the clauses in-place (as opposed to removing and re-inserting them in New) would compromise the completeness of the algorithm. Actual examples are long, depend heavily on the order in which clauses are dequeued from Passive and the redundancy elimination employed, and are, therefore, not worth exhibiting. For an intuition, it suffices to observe that by rewriting a clause, its maximum formula might change, breaking the invariant that says that every possible inference between clauses in Active has been performed.

One may also wonder if after processing a unit equality $@_i j$ it is safe to simply discard it instead of moving it to Active. After all, the sole idea of unit rewriting is to eliminate every occurrence of i. The answer, again, is that this is not avoidable. There are several reasons for this. First, one will need to take into account this clause if building a model for the initial formula from a saturated set. Second, suppose the unit clause is of the form $@_{\epsilon\langle i,r,\varphi\rangle} j$; even if we remove every occurrence of $\epsilon\langle i, r, \varphi\rangle$ by rewriting, new instances can later appear via applications of rules $\langle r\rangle_\epsilon$ and $\mathrm{PAR}^{@}_{\Diamond\epsilon}$. This means $@_{\epsilon\langle i,r,\varphi\rangle} j$ must be still considered for inferences.

We have found two further complications implementing unit rewriting. First, suppose the given clause is a unit equality $C_1 : @_i j$ and the clause set contains the (unit) clause $C_2 : @_i\langle r\rangle\epsilon\langle i, r, \varphi\rangle$. Rewriting C_2 using C_1 corresponds to the inference:

$$\frac{@_i\langle r\rangle\epsilon\langle i, r, \varphi\rangle \quad @_i j}{@_j\langle r\rangle\epsilon\langle i, r, \varphi\rangle}$$

But this is not a valid instance of rule $\mathrm{PAR}^{@}_{\Diamond\epsilon}$. The correct instance would be:

$$\frac{@_i\langle r\rangle\epsilon\langle i, r, \varphi\rangle \quad @_i j}{@_j\langle r\rangle\epsilon\langle i, r, \varphi\rangle \quad @_{\epsilon\langle i,r,\varphi\rangle}\epsilon\langle j, r, \varphi\rangle}$$

In fact, what we would need is to introduce, for every r and every φ occurring in the clause set, a clause of the form $@_{\epsilon\langle i,r,\varphi\rangle}\epsilon\langle j, r, \varphi\rangle$. This approach seems a little impractical, though. What we have done, instead, is perform the rewriting also *inside* ϵ-terms. For example, for clauses C_1 and C_2 above, we would generate clause $@_j\langle r\rangle\epsilon\langle j, r, \varphi\rangle$. This rewriting is clearly sound but we have not attempted to formally show that it preserves completeness yet. We were not able to find any counterexample either, after extensive testing.

For the second complication, suppose Passive contains unit equalities $C_1 : @_i j$ and $C_2 : @_i k$ with $i \succ j \succ k$. The following is not a valid instance of $\mathrm{PAR}^{@}_{\not\varphi}$:

$$\frac{@_i k \quad @_i j}{@_j k}$$

This is because $k \succ j$ is not the case, which violates the side condition of $\mathrm{PAR}^{@}_{\not\approx}$. But this means that if C_1 is selected as given and C_2 is rewritten then we might be skipping an inference using $\mathrm{PAR}^{@}_{\not\approx}$ and this might compromise completeness. We have opted to be conservative, and avoid the rewriting of C_2 using C_1 altogether.

9.4 Support for the converse modality

The *converse modality* $[\overline{r}]$ of a modality $[r]$ is defined by extending Definition 1.7 with the following clause:

$$\mathcal{M}, w \models [\overline{r}]\varphi \text{ iff } (v, w) \in R(r) \text{ implies } \mathcal{M}, v \models \varphi, \text{ for every } v \in W$$

This is a classical modal operator also known as the *past* modality in temporal logics. We have included experimental support in HyLoRes for the converse modality, in order to initially assess the modularity of the ordered calculus.

In order to extend $\mathsf{DRL}\epsilon^{\succ}_{S}$, we assume that both r and $\overline{r}$ are elements of Rel, that $\overline{\overline{r}}$ denotes r and add the following rule to the calculus:

$$[\overline{r}]\ \frac{C \vee @_m[\overline{r}]\varphi \quad D \vee @_l\langle r\rangle m}{C \vee D \vee @_l\varphi}$$

The resulting calculus is clearly sound. Moreover, since the consequent of rule $[\overline{r}]$ is smaller than its main premise, it is fairly easy to extend the proof of Theorem 8.1 to this calculus (one need to introduce a slight modification in the candidate model construction in order to guarantee that $@_i\langle r\rangle j$ and $@_j\langle\overline{r}\rangle j$ are equivalent formulas) and show that it has the reduction property for counterexamples.

Interestingly, one can repeat the proofs of Theorem 8.4, Lemma 8.4 and Theorem 8.5 almost verbatim: the case for rule $[\overline{r}]$ on each induction is analogous to the one for $[r]$. With this, one can show that HyLoRes is a decision procedure for $\mathcal{H}(@, [\overline{r}])$ (that is, $\mathcal{H}(@)$ extended with the converse modality).

Part IV

Coinduction, extractability, normal forms

Chapter 10

Modal semantics based on coinductive models

In Chapter 11 we will investigate certain normal forms for modal logics. To be a little more precise, our aim is to associate a normal form to *each modal logic*, the details of the normal form will vary depending on the interrelation of the modalities occurring in each logic. Needless to say, we are looking for a very general result. This poses a problem on its own: how can we *express* it in a way that is sufficiently general?

Before trying to answer this question, let's see why this may be a problem in the first place. For this, consider, on the one hand, logics such as $\mathcal{ML}(4)$ and $\mathcal{ML}(\mathit{Alt}_1)$, that is, $\mathcal{ML}$ restricted to models where r is, respectively, a transitive relation and a partial function, and, on the other, the hybrid logic $\mathcal{H}(@, \downarrow)$. All of these are modal logics and more expressive than the basic modal logic $\mathcal{ML}$, in the sense that the set of tautologies of each logic extends the set of tautologies of $\mathcal{ML}$. There is some important difference between $\mathcal{H}(@, \downarrow)$ and the first two logics, though; the kind of difference that make people refer to hybrid logics as "extensions of standard modal logics" [Areces and ten Cate, 2006] instead of simply "modal logics".

Both $\mathcal{ML}(4)$ and $\mathcal{ML}(\mathit{Alt}_1)$ are particular cases of $\mathcal{ML}$. A model for $\mathcal{ML}(4)$ (or for that matter $\mathcal{ML}(\mathit{Alt}_1)$) is also a model for $\mathcal{ML}$. Once we fix such a model, it makes no difference (from the point of view of semantics) if we are working in $\mathcal{ML}(4)$ or in $\mathcal{ML}$.

We defined Kripke models and hybrid models to be different objects (cf. Definition 1.5 and 1.9), but this is really only a matter of presentation.[1] The important difference is in the semantics of $\mathcal{H}(@, \downarrow)$, in that it *extends* that of $\mathcal{ML}$ with new clauses, namely those that give formal meaning to $@_i$ and $\downarrow i$ (cf. Definitions 1.7 and 1.10).

[1]Hybrid models are frequently defined as Kripke models $\langle W, R, V\rangle$ where $V(i)$ is required to be a singleton set when i is nominal. Nominals in this case are treated as a sort of proposition symbols.

Being an "extension of $\mathcal{ML}$" means that there are many result that hybrid logics cannot directly inherit from standard modal logics; instead these must be proven again in the hybrid setting. Bisimulations constitute a typical example of this: the notion of bisimulation for $\mathcal{ML}$ is not appropriate for $\mathcal{H}(@)$ and has to be adapted to account for the semantics of $@_i$ (cf. Definitions 1.11 and 1.13); this in turn means we need two different *Invariance under bisimulations* theorems (cf. Theorems 1.1 and 1.2) with slightly different proofs.

Now, $\mathcal{H}(@,\downarrow)$ contains at its core the basic modal logic $\mathcal{ML}$. Therefore, general results proven using the semantics of $\mathcal{H}(@,\downarrow)$ can indeed be extrapolated to $\mathcal{ML}$ and, hence, to logics such as $\mathcal{ML}(4)$, $\mathcal{ML}(\mathit{Alt}_1)$, etc. If $\mathcal{H}(@,\downarrow)$ were the only direction in which $\mathcal{ML}$ could be extended, this would be fine; but it is clearly not. Moreover, we already know other "extensions of standard modal logics" which would benefit from the results in Chapter 11 and that are not particular cases of $\mathcal{H}(@,\downarrow)$. One such logic is presented in the following example.

Example 10.1 (Memory logics)**.** Motivated by the search of binding operators that are somehow weaker than $\downarrow$, Areces et al. [2008] introduce the family of so-called *memory logics*. Intuitively, memory logics are modal logics whose semantics is specified in terms of Kripke models enriched with additional *data structures* to represent *memory*. The logical language is extended with a collection of operations to access and modify the data structures.

In the simplest case, a Kripke model $\mathcal{M}$ is paired with a set $S \subseteq |\mathcal{M}|$ which can be interpreted as a set of states that are "known" to us, and which represent the current "memory" of the model. We can now define the following operators ($\models_M$ extends $\models_K$ for the basic modal case):

$$\begin{array}{ll}
\langle \mathcal{M}, S\rangle, w \models_M \textcircled{k} & \text{iff } w \in S,\\
\langle \mathcal{M}, S\rangle, w \models_M \textcircled{r}\varphi & \text{iff } \langle \mathcal{M}, S \cup \{w\}\rangle, w \models_M \varphi,\\
\langle \mathcal{M}, S\rangle, w \models_M \textcircled{f}\varphi & \text{iff } \langle \mathcal{M}, S \setminus \{w\}\rangle, w \models_M \varphi,\\
\langle \mathcal{M}, S\rangle, w \models_M \textcircled{e}\varphi & \text{iff } \langle \mathcal{M}, \emptyset\rangle \models_M \varphi.
\end{array}$$

The *remember* operator $\textcircled{r}$ is a unary modality that marks the current state as being "known" or "already visited" by storing it in the "memory" S. Similarly, the operator $\textcircled{f}$ "forgets" the current state by removing it from S. The *erase* operator $\textcircled{e}$ wipes out the whole memory. These are the operators used to update the memory. The zero-ary operator $\textcircled{k}$ (read *known*) queries S to check if the current state has already been visited. Observe that $\textcircled{r}$, $\textcircled{k}$ and $\textcircled{e}$ can be seen as *binding* the instances of $\textcircled{k}$ occurring under their scope.

In the following sections we will introduce a novel semantics for the basic modal logic $\mathcal{ML}$. The distinguishing feature is the use of coinductively defined models that can be seen as a generalization of Kripke models. Standard modal logics and extensions such as hybrid logics and memory logics will fit in this framework in a natural way.

10.1 The coinductive framework

In order to describe the new framework, we will now have to redefine some notation and terminology that was already introduced in Chapter 2 We begin by redefining the modal language we will be using. A modal signature $\mathcal{S}$ will now be a pair $\mathcal{S} = \langle \mathsf{Atom}, \mathsf{Mod} \rangle$ where Atom and Mod are two countable, disjoint sets. We will use $a, b, \ldots$ to refer to elements in Atom and $m, n, \ldots$ for modal symbols in Mod. We reserve Prop, Nom and Rel for later use.

Definition 10.1 (Modal formula)**.** The set of modal formulas over the signature $\mathcal{S} = \langle \mathsf{Atom}, \mathsf{Mod} \rangle$ is defined as:

$$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid [m]\varphi,$$

where $a \in \mathsf{Atom}$ and $m \in \mathsf{Mod}$. Again, $\top$ and $\bot$ will stand for an arbitrary tautology and contradiction, respectively. We will use classical connectives such as $\wedge, \rightarrow$ and $\langle m \rangle$, taken to be defined in the usual way.

We will write $\varphi(\psi)$ to indicate that ψ is a subformula of φ and use $\varphi(\psi/\theta)$ to denote the formula obtained by uniformly substituting all appearances of ψ in φ by θ.

The language we just defined is exactly the same language of the basic modal logic $\mathcal{ML}$. Interestingly, we will be able to cast "extensions" of modal logics containing binders and operators (e.g., those in hybrid and memory logics) into this same language, in a very natural way. How we do this will become clear once we provide our definition of models.

Recall that a *pointed* Kripke model is a tuple $\langle \mathcal{M}, w \rangle$ where $\mathcal{M}$ is a (standard) Kripke model and $w \in |\mathcal{M}|$ (cf. Definition 1.9). That is, in a pointed model the "point of evaluation" is part of the (pointed) model. For convenience, throughout this chapter we will restrict our attention to pointed models. In particular, the models we are about to define will be pointed too.

The main difference between a classical pointed Kripke models and the ones we will introduce next is in the way relation symbols are interpreted. In a Kripke model every relation symbol is interpreted as a binary relation over the domain of the model or, what is equivalent, as a function that assigns to each element a subset of the domain, corresponding to the set of its successors. In our case, relations will be interpreted as functions that assign, to each element of the domain a *set of models* (over the same domain). Since the successors of an element will be models instead of other elements, our definition of models will have to be coinductive.

Definition 10.2 (Coinductive models)**.** Let $\mathcal{S} = \langle \mathsf{Atom}, \mathsf{Mod} \rangle$ be a modal signature and W be a fixed, non-empty set. $\mathbf{Mods}_W$, the class of all models with domain W (over signature $\mathcal{S}$) is the set of tuples $\langle w, W, V, R \rangle$ such that:

$$\begin{array}{rcll} w & \in & W, & \\ V(a) & \subseteq & W & \text{for } a \in \mathsf{Atom}, \\ R(m, w) & \subseteq & \mathbf{Mods}_W & \text{for } m \in \mathsf{Mod} \text{ and } w \in W. \end{array}$$

Mods is the class of all models over all domains, i.e., $\mathbf{Mods} \overset{def}{=} \bigcup_W \mathbf{Mods}_W$.

We will keep traditional practice and call w the *point of evaluation*, W the *domain*, V the *valuation*, and R the *accessibility relation*. For $\mathcal{M}$ an arbitrary model we will often write $|\mathcal{M}|$ for its domain, $w^{\mathcal{M}}$ for its point of evaluation, $V^{\mathcal{M}}$ for its valuation and $R^{\mathcal{M}}$ for its accessibility relation. We will sometimes write $\mathit{succs}^{\mathcal{M}}(m)$ for the set $R^{\mathcal{M}}(m, w^{\mathcal{M}})$ of immediate m-successors of $w^{\mathcal{M}}$.

Observe that for each W, $\mathbf{Mods}_W$ is well-defined (coinductively), and so is **Mods**, the class of all models. Being the class of all possible models, **Mods** enjoys some nice closure properties that will be useful when considering subclasses of **Mods**. In particular, we will confine our attention to model classes which are *closed under accessibility relations*.

Definition 10.3 (Extension of a model)**.** Given $\mathcal{M} \in \mathbf{Mods}_W$, let $\mathrm{Ext}(\mathcal{M})$, the *extension of* $\mathcal{M}$, be the smallest subset of $\mathbf{Mods}_W$ that contains $\mathcal{M}$ and is such that if $\mathcal{N} \in \mathrm{Ext}(\mathcal{M})$, then $R^{\mathcal{N}}(m, v) \subseteq \mathrm{Ext}(\mathcal{M})$ for all $m \in \mathsf{Mod}$, $v \in W$.

Definition 10.4 (Closed class)**.** A non-empty class of models $\mathcal{C}$ is *closed under accessibility relations* (we will say that $\mathcal{C}$ is a *closed class*, for short) whenever $\mathcal{M} \in \mathcal{C}$ implies $\mathrm{Ext}(\mathcal{M}) \subseteq \mathcal{C}$.

In other words, the extension of a model is simply the class of models reachable via the transitive closure of the union of its accessibility relations; and a class of models $\mathcal{C}$ is closed if for every model $\mathcal{M} \in \mathcal{C}$ the extension of $\mathcal{M}$ is also included in $\mathcal{C}$. Clearly, **Mods** is a closed class and, as we will discuss below, it seems natural to restrict ourselves to investigate only closed classes.

Having properly defined what models are, the definition of the satisfiability relation $\models$ is straightforward:

Definition 10.5 (Semantics)**.** For each $\mathcal{M} = \langle w, W, V, R \rangle$ in **Mods** we define:

$$\begin{array}{lll} \mathcal{M} \models a & \text{iff} & w \in V(a) \\ \mathcal{M} \models \neg\varphi & \text{iff} & \mathcal{M} \not\models \varphi \\ \mathcal{M} \models \varphi \vee \psi & \text{iff} & \mathcal{M} \models \varphi \text{ or } \mathcal{M} \models \psi \\ \mathcal{M} \models [m]\varphi & \text{iff} & \mathcal{M}' \models \varphi, \text{ for all } \mathcal{M}' \in R(m, w). \end{array}$$

If $\mathcal{C}$ is a closed class, we write $\mathcal{C} \models \varphi$ whenever $\mathcal{M} \models \varphi$ for every $\mathcal{M}$ in $\mathcal{C}$, and we say that $\Gamma_{\mathcal{C}} = \{\varphi \mid \mathcal{C} \models \varphi\}$ is the *logic* defined by $\mathcal{C}$.

It is instructive to compare Definitions 1.10 and 10.5. If we abstract away the details on how a model is represented in each case, we can say that they are roughly equivalent. But if we restrict ourselves to the appropriate class of models, we can actually ensure that $[m]$ behaves in very different ways. Let us see an example.

Example 10.2 (The universal modality A)**.** Given a Kripke model $\mathcal{M}$ with domain W the usual semantic clause for the universal modality A would be

$$\mathcal{M}, w \models \mathsf{A}\varphi \text{ iff } \mathcal{M}, w' \models \varphi, \text{ for all } w' \in W.$$

Instead, let $\mathcal{S} = \langle \mathsf{Atom}, \mathsf{Mod} \rangle$ with $\mathsf{A} \in \mathsf{Mod}$, and let $\mathcal{C}_{\mathsf{A}}$ be the largest class of models in this signature such that

$$\text{if } \mathcal{M} \in \mathcal{C}_{\mathsf{A}}, \text{ then } R^{\mathcal{M}}(\mathsf{A}, w) = \{\langle w', |\mathcal{M}|, V^{\mathcal{M}}, R^{\mathcal{M}} \rangle \mid w' \in |\mathcal{M}|\}.$$

That is, the A-successors of w are those models identical to $\mathcal{M}$ except in that their point of evaluation is an arbitrary element of the domain. Clearly, the semantic condition of $[\mathsf{A}]$ in $\mathcal{C}_{\mathsf{A}}$ (as given in Definition 10.5) coincides exactly with the semantic definition of the universal modality A over standard Kripke models.

By taking suitable classes of models, we can naturally capture many different modal operators. This follows in spirit classical modal semantics, where one obtains a richer logic by considering certain classes of Kripke models. In fact, already in the classical case, if one considers the class of Kripke models where the relational symbol A is interpreted as a total relation (i.e., those models where $\forall xy.A(x,y)$ holds), one obtains the logic $\mathcal{C}_{\mathsf{A}}$ of Example 10.2. What we will see in Section 10.2 is that in this coinductive setting we can also capture this way binders and other *non-relational* modal operators.

When defining classes of models it is necessary to require the classes to be closed (cf. Definition 10.4). If a class $\mathcal{C}$ is not closed, the evaluation of some modal formulas on a model in $\mathcal{C}$ might require the inspections of models which are outside the class. Moreover, as it follows from Proposition 10.1 below, every closed class induces a *normal* modal logic [Blackburn et al., 2002]. In the rest of this thesis we will usually implicitly assume that **every class is closed** and that all its models conform to some particular, but arbitrary, signature. We will only mention these conditions explicitly for additional emphasis.

Proposition 10.1. *Let $\mathcal{C}$ be a (closed) class. Then*

i. $\Gamma_{\mathcal{C}}$ contains all instances of propositional tautologies.

ii. $\Gamma_{\mathcal{C}}$ *is closed under modus-ponens.*

iii. $\Gamma_{\mathcal{C}}$ *is closed under necessitation, i.e., if* $\varphi \in \Gamma_{\mathcal{C}}$*,* $[m]\varphi \in \Gamma_{\mathcal{C}}$*.*

iv. $\Gamma_{\mathcal{C}}$ *contains every instance of axiom* K_m*, i.e., for every formula* φ *and all* $m \in \mathit{Mod}$*,* $[m](\varphi \to \psi) \to ([m]\varphi \to [m]\psi) \in \Gamma_{\mathcal{C}}$*.*

Proof. Closure under tautologies and modus-ponens follows trivially from the semantics of the boolean operators.

Necessitation is straightforward: suppose $\varphi \in \Gamma_{\mathcal{C}}$, then $\mathcal{M} \models \varphi$ for all $\mathcal{M} \in \mathcal{C}$, but since $\mathcal{C}$ is a closed class this implies $\mathcal{M} \models [m]\varphi$ and consequently $[m]\varphi \in \Gamma_{\mathcal{C}}$. (This is the only case were we need to assume that $\mathcal{C}$ is closed).

Finally, let $[m](\varphi \to \psi) \to ([m]\varphi \to [m]\psi)$ be an instance of axiom K_m and suppose $\mathcal{M} \models [m](\varphi \to \psi)$ and $\mathcal{M} \models [m]\varphi$, for some $\mathcal{M} \in \mathcal{C}$. It follows that for every $\mathcal{M}' \in R(m, w^{\mathcal{M}})$, $\mathcal{M}' \models \varphi \to \psi$ and $\mathcal{M}' \models \varphi$ both hold and, therefore, $\mathcal{M}' \models \psi$ holds too. Hence, $\mathcal{M} \models [m]\psi$. □

By Proposition 10.1 then, the minimal logic (generated by the class of all possible models) $\Gamma_{\mathbf{Mods}}$ is a normal modal logic. As we will show in Proposition 10.2 below, it coincides with the basic modal logic K (cf. Definition 1.10). In particular, $\Gamma_{\mathbf{Mods}}$ is closed under uniform substitution, that is:

$$\text{if } \varphi \in \Gamma_{\mathbf{Mods}} \text{ then } \varphi(a/\psi) \in \Gamma_{\mathbf{Mods}} \text{ for } a \in \mathsf{Atom}.$$

But for an arbitrary $\mathcal{C}$, $\Gamma_{\mathcal{C}}$ doesn't need to be closed under uniform substitution (we will see some examples of this in the following section).

We are interested in defining subclasses of **Mods**, in particular, classes that are closed under accessibility relations. To simplify definitions we introduce the following piece of notation.

Definition 10.6 (Defining conditions)**.** Predicate P is a *defining condition* for $\mathcal{C}$ if $\mathcal{C}$ is the *largest* class such that $\mathcal{M} \in \mathcal{C}$ implies that $P(\mathcal{M})$ holds.

Observe that if P is a defining condition for $\mathcal{C}$, then $\mathcal{C}$ is necessarily a closed class. We can use this notation to properly define the class of models where the accessibility relations are standard relational modalities.

Definition 10.7 (Relational modalities: the classes $\mathcal{C}^{\mathsf{K}}_m$ and $\mathcal{C}^{\mathsf{K}}$)**.** For each $m \in \mathsf{Mod}$, let $\mathcal{C}^{\mathsf{K}}_m$ be the class defined by the following defining condition:

$$P^{\mathsf{K}}_m(\mathcal{M}) \iff \forall v \in |\mathcal{M}|, R^{\mathcal{M}}(m, v) \subseteq \{\langle v', |\mathcal{M}|, V^{\mathcal{M}}, R^{\mathcal{M}}\rangle \mid v' \in |\mathcal{M}|\}$$

Observe that P^{K}_m is true of a model $\mathcal{M}$ if every successor of $w^{\mathcal{M}}$ is identical to $\mathcal{M}$ except perhaps on its point of evaluation. We will call m a *relational modality* when it is interpreted in $\mathcal{C}^{\mathsf{K}}_m$. Define the class of models $\mathcal{C}^{\mathsf{K}}$ over the signature $\mathcal{S} = \langle \mathsf{Atom}, \mathsf{Mod}\rangle$ as follows: $\mathcal{M} \in \mathcal{C}^{\mathsf{K}}$ iff for every modality $m \in \mathsf{Mod}$, $\mathcal{M} \in \mathcal{C}^{\mathsf{K}}_m$. That is, all modalities are interpreted in $\mathcal{C}^{\mathsf{K}}$ as relational modalities.

Proposition 10.2. *Let $\mathcal{S} = \langle \mathsf{Atom}, \mathit{Mod} \rangle$. The following properties hold:*

i. $\mathcal{C}^{\mathsf{K}} \subset \mathbf{Mods}$.

ii. $\Gamma_{\mathcal{C}^{\mathsf{K}}}$ *coincides with the basic modal logic* K *(over signature* $\mathcal{S}$*).*

iii. $\Gamma_{\mathbf{Mods}}$ *coincides with the basic modal logic* K *(over signature* $\mathcal{S}$*).*

Proof. i. It should be clear that $\mathcal{C}^{\mathsf{K}}$ is a strict subset of $\mathbf{Mods}$. Take for example a signature $\langle \{p\}, \{m\} \rangle$ and consider models $\mathcal{M}$ and $\mathcal{N}$ in $\mathbf{Mods}$ where

$$\mathcal{M} = \langle a, \{a\}, \{(m, a) \mapsto \{\mathcal{N}\}\}, \{p \mapsto \{a\}\} \rangle$$
$$\mathcal{N} = \langle a, \{a\}, \{(m, a) \mapsto \{\mathcal{M}\}\}, \{p \mapsto \emptyset\} \rangle.$$

As $\mathcal{M} \in \mathbf{Mods}$ but $\mathcal{M} \notin \mathcal{C}^{\mathsf{K}}$, we have that $\mathcal{C}^{\mathsf{K}} \subset \mathbf{Mods}$.

ii. We will use the well-known fact that K is complete with respect to the class of all (classical) pointed Kripke models. Now, to every $\mathcal{M} \in \mathcal{C}^{\mathsf{K}}$ we associate a unique pointed Kripke model $f(\mathcal{M}) = \langle \langle W^{\mathcal{M}}, R, V^{\mathcal{M}} \rangle, w^{\mathcal{M}} \rangle$, where $R(m, u) = \{v \mid \langle v, W^{\mathcal{M}}, R^{\mathcal{M}}, V^{\mathcal{M}} \rangle \in R^{\mathcal{M}}(m, u)\}$.

Clearly, f is a bijection between $\mathcal{C}^{\mathsf{K}}$ and the class of all pointed Kripke models, such that for every $\mathcal{M} \in \mathcal{C}^{\mathsf{K}}$, $\mathcal{M} \models \varphi$ iff $f(\mathcal{M}) \models_{\mathsf{K}} \varphi$. Therefore, $\Gamma_{\mathcal{C}^{\mathsf{K}}} = \mathsf{K}$.

iii. Since $\mathcal{C}^{\mathsf{K}} \subset \mathbf{Mods}$ we may conclude $\Gamma_{\mathbf{Mods}} \subseteq \Gamma_{\mathcal{C}^{\mathsf{K}}}$. By Proposition 10.1 we conclude $\mathsf{K} \subseteq \Gamma_{\mathbf{Mods}} \subseteq \mathsf{K}$. □

Proposition 10.2 shows that the basic modal logic K can be recast (in two different ways) using the new semantics. The same can be done for many other modal languages, and we will show some examples in the next section, focusing on hybrid and hybrid-related languages.

10.2 Hybrid logics as classes of models

Consider again the sets Prop, Nom and Rel from Chapter 1, and define with them a "hybrid" signature $\mathcal{S} = \langle \mathsf{Atom}, \mathsf{Mod} \rangle$ where

$$\mathsf{Atom} = \mathsf{Prop} \cup \mathsf{Nom} \cup \{ⓚ\},$$
$$\mathsf{Mod} = \mathsf{Rel} \cup \{\mathsf{A}\} \cup \{@_i \mid i \in \mathsf{Nom}\} \cup \{{\downarrow}i \mid i \in \mathsf{Nom}\} \cup \{ⓡ, ⓕ, ⓔ\}.$$

In what follows, we will work with formulas from the language of Definition 10.1 over either $\mathcal{S}$ or over some $\mathcal{S}'$ contained in $\mathcal{S}$.

In Figure 10.1 we present several closed classes via their defining conditions. These conditions are of different kind and, in general, give the expected semantics to a modal symbol. The only exception is $\mathcal{C}_a$ which can

Class	Defining condition
$\mathcal{C}_a$	$\mathcal{P}_a(\mathcal{M}) \iff V(a)$ is a singleton
$\mathcal{C}_{\mathsf{A}}$	$\mathcal{P}_{\mathsf{A}}(\mathcal{M}) \iff R(\mathsf{A}, w) = \{\langle v, W, V, R\rangle \mid v \in W\}$
$\mathcal{C}_{@_i}$	$\mathcal{P}_{@_i}(\mathcal{M}) \iff R(@_i, w) = \{\langle v, W, V, R\rangle \mid v \in V(i)\}$
$\mathcal{C}_{\downarrow i}$	$\mathcal{P}_{\downarrow i}(\mathcal{M}) \iff R(\downarrow i, w) = \{\langle w, W, V[i \mapsto \{w\}], R\rangle\}$
$\mathcal{C}_{\textcircled{r}}$	$\mathcal{P}_{\textcircled{r}}(\mathcal{M}) \iff R(\textcircled{r}, w) = \{\langle w, W, V[\textcircled{k} \mapsto V(\textcircled{k}) \cup \{w\}], R\rangle\}$
$\mathcal{C}_{\textcircled{f}}$	$\mathcal{P}_{\textcircled{f}}(\mathcal{M}) \iff R(\textcircled{f}, w) = \{\langle w, W, V[\textcircled{k} \mapsto V(\textcircled{k}) \setminus \{w\}], R\rangle\}$
$\mathcal{C}_{\textcircled{e}}$	$\mathcal{P}_{\textcircled{e}}(\mathcal{M}) \iff R(\textcircled{e}, w) = \{\langle w, W, V[\textcircled{k} \mapsto \emptyset], R\rangle\}$

Figure 10.1: Several classes and their defining conditions. It is assumed that $\mathcal{M} = \langle w, W, V, R\rangle$.

be described as *the class of models where atom a is, semantically, a nominal.* For distinct a and b, $\mathcal{C}_a$ and $\mathcal{C}_b$ are, of course, distinct classes. Observe that uniform substitution fails in $\mathcal{C}_a$: $(a \wedge b \wedge \langle m\rangle a) \rightarrow \langle m\rangle b \in \Gamma_{\mathcal{C}_a}$ but $(\top \wedge b \wedge \langle m\rangle \top) \rightarrow \langle m\rangle b \notin \Gamma_{\mathcal{C}_a}$.

The other classes in Figure 10.1 are defined by imposing conditions on the accessibility relation. Consider, first, $\mathcal{C}_{\mathsf{A}}$ and $\mathcal{C}_{@_i}$; the former was already discussed in Example 10.2, the latter captures the "jump-to-i" meaning of the $@_i$ operator of hybrid logics but in a slightly generalized way: i need not be a nominal (i.e., interpreted as a singleton) in $\mathcal{C}_{@_i}$.

Both $\mathcal{C}_{\mathsf{A}}$ and $\mathcal{C}_{@_i}$ share a common feature: they are relational modalities, that is, $\mathcal{C}_{\mathsf{A}} \subseteq \mathcal{C}^{\mathsf{K}}_{\mathsf{A}}$ and $\mathcal{C}_{@_i} \subseteq \mathcal{C}^{\mathsf{K}}_{@_i}$. This is just another way to state the fact that the semantics of the universal modality A, and satisfiability operators $@_i$ of hybrid logics can all be captured on Kripke models by restricting evaluation to the class of models where the relation that interprets them is, respectively, the total relation and the "point-to-i" relation.

On the other hand, $\mathcal{C}_{\downarrow i}$, the class of models where $[\downarrow i]$ *behaves like the* $\downarrow i$ *operator of hybrid logics*, is defined by imposing *conditions on the valuation* of the accessible models: "there is only one $\downarrow i$-successor of $\mathcal{M}$ and differs from $\mathcal{M}$ only in that i must be interpreted as w". This means, that $\mathcal{C}_{\downarrow i}$ is not a subclass of $\mathcal{C}^{\mathsf{K}}_{\downarrow i}$ (that is, $[\downarrow i]$ is not a relational modality in $\mathcal{C}_{\downarrow i}$). It is not surprising, then, that uniform substitution fails on this class: while it is clear that $\mathcal{C}_{\downarrow i} \models [\downarrow i]i$, the uniform substitution of atom i by p yields the formula $[\downarrow i]p$ which is not $\mathcal{C}_{\downarrow i}$-valid.

Finally, classes $\mathcal{C}_{\textcircled{r}}$, $\mathcal{C}_{\textcircled{f}}$ and $\mathcal{C}_{\textcircled{e}}$ define the classes associated to the operators of memory logics presented in Example 10.1. It is interesting to compare their defining conditions with that of $\mathcal{C}_{\downarrow i}$. It becomes quite evident, then, that these memory operators bind atom $\textcircled{k}$ in exactly the same way in which $[\downarrow i]$ binds i.

The classes defined in Figure 10.1 give semantics to isolated operators

$$\begin{aligned}
\mathcal{C}_{\mathsf{Nom}} &\overset{def}{=} \bigcap_{i\in\mathsf{Nom}} \mathcal{C}_i \\
\mathcal{C}_{\mathsf{Rel}} &\overset{def}{=} \bigcap_{m\in\mathsf{Rel}} \mathcal{C}^{\mathsf{K}}_m \\
\mathcal{C}_{@} &\overset{def}{=} \bigcap_{i\in\mathsf{Nom}} \mathcal{C}_{@_i} \\
\mathcal{C}_{\downarrow} &\overset{def}{=} \bigcap_{i\in\mathsf{Nom}} \mathcal{C}_{\downarrow i} \\
\mathcal{C}_{\mathcal{H}(@,\downarrow)} &\overset{def}{=} \mathcal{C}_{\mathsf{Nom}} \cap \mathcal{C}_{\mathsf{Rel}} \cap \mathcal{C}_{@} \cap \mathcal{C}_{\downarrow}
\end{aligned}$$

Figure 10.2: Closed classes defined as intersections of other classes.

and atoms. In order to define a logic like $\mathcal{H}(@,\downarrow)$ we need to *combine* this classes. Now, observe that if $\mathcal{C}$ and $\mathcal{C}'$ are closed classes, then the intersection of both classes is also a (perhaps empty) closed class.

Figure 10.2 exhibit several classes defined in this way. For instance, $\mathcal{C}_{\mathsf{Nom}}$ is the class of models where every atom i in Nom has the semantics of a nominal (true at exactly one point of the model) while $\mathcal{C}_{\mathsf{Rel}}$ is the class where every modality in Rel behaves like a relational modality (recall that we are working with a "hybrid" signature $\mathcal{S}$, where $\mathsf{Nom} \subset \mathsf{Atom}$ and $\mathsf{Rel} \subset \mathsf{Mod}$). Similarly, $\mathcal{C}_{\downarrow}$ and $\mathcal{C}_{@}$ are the classes of models where, for every $i \in \mathsf{Nom}$, $[@_i]$ and $[\downarrow i]$ behave like (a generalization of) the hybrid operators. Finally, $\mathcal{C}_{\mathcal{H}(@,\downarrow)}$ captures the hybrid logic $\mathcal{H}(@,\downarrow)$.

It is worth stressing one point: in the class $\mathcal{C}_{@}$, the $[@_i]$ operator behaves slightly different than the hybrid operator $@_i$ of $\mathcal{H}(@)$. For example, $\mathcal{C}_{@} \not\models [@_i]\varphi \leftrightarrow \langle @_i\rangle\varphi$; that is $@_i$ is not self dual. But if we force every nominal i to be true at exactly one point of the domain, then every $@_i$ becomes self-dual again; i.e., $\mathcal{C}_{\mathsf{Nom}} \cap \mathcal{C}_{@} \models [@_i]\varphi \leftrightarrow \langle @_i\rangle\varphi$.

Unless otherwise stated, from here on we assume that all operators for which we introduced a special notation (i.e., A, $@_i$, $\downarrow i$, ⓚ, ⓡ, ⓕ, ⓔ and nominals) are interpreted on classes of models that satisfy the corresponding defining conditions introduced in this section.

10.3 Coinductive models and bisimulations

We will end this chapter revisiting the notion of bisimulation, which is central in modal model theory. Also here, the coinductive take on models brings interesting surprises.

Definition 10.8 (Bisimulations)**.** Let $\mathcal{M}$ and $\mathcal{M}'$ be two models. A *bisimulation* between $\mathcal{M}$ and $\mathcal{M}'$ is a relation $Z \subseteq \mathrm{Ext}(\mathcal{M}) \times \mathrm{Ext}(\mathcal{M}')$ such that $(\mathcal{M}, \mathcal{M}') \in Z$ and if $(\langle w, W, V, R\rangle, \langle w', W', V', R'\rangle) \in Z$ then:

(atom) $w \in V(a)$ iff $w' \in V'(a)$, for all $a \in \mathsf{Atom}$,

(zig) $\mathcal{N} \in R(m, w)$ implies $(\mathcal{N}, \mathcal{N}') \in Z$ for some $\mathcal{N}' \in R'(m, w')$,

(zag) $\mathcal{N}' \in R'(m, w')$ implies $(\mathcal{N}, \mathcal{N}') \in Z$ for some $\mathcal{N} \in R(m, w)$.

When there exists a bisimulation Z between $\mathcal{M}$ and $\mathcal{M}'$ we say that $\mathcal{M}$ *and* $\mathcal{M}'$ *bisimilar*, notated $\mathcal{M} \underline{\leftrightarrow} \mathcal{M}'$.

The classic result of invariance of modal formulas under bisimulation we reviewed in Chapter 1 (cf. Theorem 1.1) can easily be proved in this setting.

Theorem 10.1. *If $\mathcal{M} \underline{\leftrightarrow} \mathcal{M}'$, then $\mathcal{M} \models \varphi$ iff $\mathcal{M}' \models \varphi$, for all φ.*

The interesting thing to observe is that this very general notion of bisimulation works for every modal logic definable as a closed subclass of **Mods**. In other words, this definition is capturing a variety of notions of bisimulation. Many well known bisimulations can be seen as specializations of Definition 10.8. We illustrate this with an example.

Example 10.3. We discussed in Chapter 1 that the adequate notion of bisimulation for the hybrid logic $\mathcal{H}(@)$ requires agreement of models on all named worlds (cf. Definition 1.13). Consider now the closed class

$$\mathcal{C}_{\mathcal{H}(@)} \overset{def}{=} \mathcal{C}_{\mathsf{Nom}} \cap \mathcal{C}_{\mathsf{Rel}} \cap \mathcal{C}_{@} \tag{10.1}$$

which corresponds (by an argument similar to the one used in the proof of Proposition 10.2) to the class of (pointed) hybrid models, and suppose we have $\mathcal{M} \underline{\leftrightarrow} \mathcal{N}$ for $\mathcal{M}, \mathcal{N} \in \mathcal{C}_{\mathcal{H}(@)}$. This means that $(\mathcal{M}, \mathcal{N}) \in Z$, for some bisimulation Z. Let $\mathcal{M}_i = \langle w, |\mathcal{M}|, V^{\mathcal{M}}, R^{\mathcal{M}} \rangle$ and $\mathcal{N}_i = \langle v, |\mathcal{N}|, V^{\mathcal{N}}, R^{\mathcal{N}} \rangle$ be two models such that $\{w\} = V^{\mathcal{M}}(i)$ and $\{v\} = V^{\mathcal{N}}(i)$. By definition, we know that:

$$\mathcal{C}_i \subseteq \mathcal{C}_{\mathsf{Nom}} \subset \mathcal{C}_{\mathcal{H}(@)}. \tag{10.2}$$

Therefore, $\mathcal{M}, \mathcal{N} \in \mathcal{C}_{\mathcal{H}(@)}$ implies $\mathcal{M}_i, \mathcal{N}_i \in \mathcal{C}_{\mathcal{H}(@)}$. Moreover, from this and the defining condition for $\mathcal{C}_{@_i}$, we may conclude $succs^{\mathcal{M}}(@_i) = \{\mathcal{M}_i\}$ and $succs^{\mathcal{N}}(@_i) = \{\mathcal{N}_i\}$. Using either **(zig)** or **(zag)**, we obtain $(\mathcal{M}_i, \mathcal{N}_i) \in Z$.

Unsurprisingly, we can also take a finer look, and adapt the notion of k-bisimulations to the present setting. Here too, Definition 10.9 will subsume not only that of k-bisimulations for the basic modal logic, but also notions such as the k-bisimulation for $\mathcal{H}(@)$ of Definition 1.15.

Definition 10.9 (k-bisimulations)**.** Given two models $\mathcal{M}$ and $\mathcal{M}'$ we say that $\mathcal{M}$ *and* $\mathcal{M}'$ *are k-bisimilar* (notation $\mathcal{M} \underline{\leftrightarrow}_k \mathcal{M}'$) if there exists a sequence of binary relations $Z_0 \supseteq Z_1 \supseteq \cdots \supseteq Z_k$ such that $(\mathcal{M}, \mathcal{M}') \in Z_k$ and for all $\mathcal{N} = \langle w, W, V, R \rangle \in \mathrm{Ext}(\mathcal{M})$ and $\mathcal{N}' = \langle w', W', V', R' \rangle \in \mathrm{Ext}(\mathcal{M}')$:

1. $(\mathcal{N}, \mathcal{N}') \in Z_0$ implies $w \in V(a)$ iff $w' \in V'(a)$, for all $a \in \mathsf{Atom}$,

2. if $(\mathcal{N}, \mathcal{N}') \in Z_{i+1}$, then
   (a) $\mathcal{N}_2 \in R(m, w)$ implies $(\mathcal{N}_2, \mathcal{N}_2') \in Z_i$ for some $\mathcal{N}_2' \in R'(m, w')$,
   (b) $\mathcal{N}_2' \in R'(m, w')$ implies $(\mathcal{N}_2, \mathcal{N}_2') \in Z_i$ for some $\mathcal{N}_2 \in R(m, w)$.

Such a sequence is called a *k-bisimulation between $\mathcal{M}$ and $\mathcal{M}'$.*

One can also prove an "invariance under k-bisimulations" theorem; notice also how the simple notion of modal depth of Definition 10.10 handles the additional cases of Definition 1.16.

Definition 10.10 (Modal depth)**.** The *modal depth* of a formula φ (notation, $md(\varphi)$) is a function from formulas to natural numbers defined as:

$$\begin{array}{lcl} md(a) & = & 0, \text{ for } a \in \mathsf{Atom} \\ md(\neg\varphi) & = & md(\varphi) \\ md(\varphi \vee \psi) & = & \max\{md(\varphi), md(\psi)\} \\ md([m]\varphi) & = & 1 + md(\varphi). \end{array}$$

Theorem 10.2 (Invariance under k-bisimulations)**.** *If $\mathcal{M} \underline{\leftrightarrow}_k \mathcal{M}'$, then, for all φ such that $md(\varphi) \leq k$, $\mathcal{M} \models \varphi$ iff $\mathcal{M}' \models \varphi$.*

We have now in place all the tools we need to tackle the main technical results of Chapter 11 and, therefore, we will stop here. If we came to an abrupt end is simply because a thorough development of this coinductive framework is out of the scope of this thesis. We reckon this view on modal models deserves further investigation and we expect to pursue it in future work.

Chapter 11

Extractability and normal forms

When working with the basic modal logic, it is standard to take the modal depth of a formula as a measure of its complexity. For example, from the tree-model property (Proposition 5.1) and the invariance under k-bisimulations theorem (Theorems 1.3 and 10.2), we know that when searching for a model of a formula φ in the basic modal logic, it is enough to consider models which are trees of depth at most the modal depth of φ. We will take a closer look at this notion, in the general set up that we presented in Chapter 10.

Consider, for example, the universal modality A (cf. Example 10.2). If Rel is a set of relational modalities (cf. Definition 10.7), then every formula in the signature $\langle \mathsf{Atom}, \mathsf{Rel} \cup \{\mathsf{A}\}\rangle$ is equivalent to a formula where the $[\mathsf{A}]$ operator only appears at modal depth zero. It suffices to verify the following equivalence:

$$\varphi([\mathsf{A}]\psi) \leftrightarrow \big([\mathsf{A}]\psi \to \varphi([\mathsf{A}]\psi/\top)\big) \wedge \big(\neg[\mathsf{A}]\psi \to \varphi([\mathsf{A}]\psi/\bot)\big) \qquad (11.1)$$

In other words, we can check $[\mathsf{A}]\psi$ once and for all at an arbitrary state in the model, and depending of the outcome replace it by $\top$ or $\bot$ throughout the formula. This kind of behavior is not particular to the universal modality; the $@_i$ operators behave in a similar way (independently even of whether i is a nominal or not). And indeed we have that, for Rel a set of relational modalities, every formula in the signature $\langle \mathsf{Atom}, \mathsf{Rel} \cup \{@_i\}\rangle$ is equivalent to a formula where $[@_i]$ only appears at modal depth zero, witness the equivalence:

$$\varphi([@_i]\psi) \leftrightarrow \big([@_i]\psi \to \varphi([@_i]\psi/\top)\big) \wedge \big(\neg[@_i]\psi \to \varphi([@_i]\psi/\bot)\big) \qquad (11.2)$$

Let us come back, then, to the idea of using modal depth as a complexity measure. Clearly, if the language contains operators which behave like $[\mathsf{A}]$ and $[@_i]$ we will have to do better than just count the maximum nesting of operators. If we know that these modalities can be 'extracted' from the scope of other operators, we should rather consider some complexity measure that takes this into account. For example, we know that every formula containing only the $[@_i]$ modality, is equivalent to a formula of modal depth one. Saying that the complexity of $[@_i]^n p$, for example, is n seems way out of line.

But deciding when a modality can be extracted is not trivial. Consider the following examples:

Example 11.1. Let $\mathcal{M} \in \mathcal{C}_{\mathcal{H}(@,\downarrow)}$ be such that $w^{\mathcal{M}} \in V^{\mathcal{M}}(p)$ and $V^{\mathcal{M}}(i) \nsubseteq V^{\mathcal{M}}(p)$. Then we have $\mathcal{M} \models [\downarrow i](p \wedge [@_i]p)$ while $\mathcal{M} \not\models \neg[@_i]p \rightarrow [\downarrow i](p \wedge \bot)$ (since $\mathcal{M} \not\models [@_i]p$) which contradicts (11.2).

Example 11.2. Let $\mathcal{C} \subseteq \mathcal{C}_{\textcircled{r}} \cap \mathcal{C}_{\mathsf{A}}$. It is easy to verify that $\mathcal{C} \models [\textcircled{r}]\neg[\mathsf{A}]\neg\textcircled{k}$. Now, for every $\mathcal{M} \in \mathcal{C}$ with $V^{\mathcal{M}}(\textcircled{k}) = \emptyset$, we have $\mathcal{M} \not\models [\mathsf{A}]\neg\textcircled{k} \rightarrow [\textcircled{r}]\neg\top$, contradicting (11.1).

In other words, 'extractability' is not a property of a single modality, but rather of the whole language interpreted in a given class of models. We will provide a definition of extractability in the next section, and investigate ways of determining when a modality can be extracted. Our final goal will be to define a notion of *extracted modal depth* that closer reflects the complexity of a formula.

11.1 Extractable Modalities

We have seen that *extractability* is not a simple notion to characterize, especially when the language contains non-relational modalities like $\downarrow i$ or $\textcircled{r}$ which can block the extraction. But what do we mean when we say that a modality can be extracted from another? Let us start by defining when a modality has *not* been extracted.

Definition 11.1 (Immediate scope)**.** Given two modalities m and n and a formula φ, we say that *n occurs in φ in the immediate scope of m* if and only if in the syntactic formation tree of φ there are two nodes e_1 and e_2 such that e_1 is labeled $[m]$, e_2 is labeled $[n]$, e_1 is an ancestor of e_2, and in the path from e_1 to e_2 there are only boolean operators.

Now, if n is $\mathcal{C}$-extractable from m, then every formula φ is equivalent in $\mathcal{C}$ to a formula where n does not occurs in the immediate scope of of m. Formally,

Definition 11.2. We say that *n is $\mathcal{C}$-extractable from m* if for any formula φ there exists a formula φ' such that n is not in the immediate scope of m in φ' and $\mathcal{C} \models \varphi \leftrightarrow \varphi'$.

Definition 11.2 does the job when the language contains only two modalities, but consider the following example:

Example 11.3. Pick a class of models $\mathcal{C}$ over signature $\mathcal{S} = \langle \mathsf{Prop}, \{m, n\}\rangle$ such that n is not $\mathcal{C}$-extractable from m (e.g., $\mathcal{C}$ can be the class of relational models over $\mathcal{S}$). Let $\mathcal{S}' = \langle \mathsf{Prop}, \{m, n, n'\}\rangle$ and let $\mathcal{C}'$ be the class of models over $\mathcal{S}'$ such that $\langle W, R', V, n\rangle \in \mathcal{C}'$ iff $\langle W, R, V, n\rangle \in \mathcal{C}$, where $R'(m, x) = R(m, x)$ and $R'(n, x) = R'(n', x) = R(n, x)$, for all $x \in W$. That is, the models of $\mathcal{C}'$ are obtained from those of $\mathcal{C}$ by interpreting n' exactly like n.

Clearly, n is $\mathcal{C}'$-extractable from m because any formula φ containing $[n]$ is equivalent in $\mathcal{C}'$ to the formula φ' obtained by replacing each appearance of $[n]$ by $[n']$ and in φ' no n is in the immediate scope of m. Symmetrically, n' is also $\mathcal{C}'$-extractable from m by replacing all appearances of $[n']$ by $[n]$. But it is not true that every formula is $\mathcal{C}'$-equivalent to one where *neither n nor n'* occur in the immediate scope of m, or otherwise n would be $\mathcal{C}$-extractable from m, which would be a contradiction.

We need then a slightly more involved definition of extractability:

Definition 11.3 ($\mathcal{C}$-extractability)**.** Let $S \subseteq \mathsf{Mod} \times \mathsf{Mod}$ be a set of pairs of modalities. We say that S *is $\mathcal{C}$-extractable* if for any formula φ there exists a formula φ' such that

1. for all $(m, n) \in S$ we have that n is not in the immediate scope of m in φ',

2. $\mathcal{C} \models \varphi \leftrightarrow \varphi'$.

If S is the singleton $\{(m, n)\}$ then S is $\mathcal{C}$-extractable if and only if n is $\mathcal{C}$-extractable from m.

Notice that the notion of $\mathcal{C}$-extractability is defined in terms of a set of pairs of modalities and for a given model class. Extractability depends not only on the modalities but on the full language (i.e., in which other operators can appear in a formula) and this is captured by restricting the definition to a given class $\mathcal{C}$.

Example 11.4. Let us consider again the signatures and classes of models of Example 11.3. While $\{(m, n)\}$ and $\{(m, n')\}$ are both $\mathcal{C}'$-extractable, $\{(m, n'), (m, n')\}$ is not $\mathcal{C}'$-extractable.

Although extractability depends on the full language, the good news is that once you know that a modality is extractable from another, the property is preserved when considering a more expressive logic. This follows directly from Definition 11.3. If m is $\mathcal{C}$-extractable from n and $\mathcal{C}' \subseteq \mathcal{C}$ then m is $\mathcal{C}'$-extractable from n too. For example, let $\mathcal{C}_1 = \mathcal{C}_{\mathsf{A}} \cap \mathcal{C}^K_m$ and assume we already proved that A is $\mathcal{C}_1$-extractable from m. If we want to add $\downarrow i$ to the language we will be working in the class $\mathcal{C}_2 = \mathcal{C}_1 \cap \mathcal{C}_{\downarrow i}$, but since $\mathcal{C}_2 \subset \mathcal{C}_1$ we still know A is $\mathcal{C}_2$-extractable from m.

In the rest of this section, we will discuss some sufficient conditions that will ensure $\mathcal{C}$-extractability.

The first sufficient condition we will mention can be motivated by the following observation: if the truth value of an arbitrary $[n]\varphi$ does not change when moving from one state to an m-successor in $\mathcal{C}$, then n should be extractable from m.

Definition 11.4 (m-invariant in $\mathcal{C}$)**.** We say that n *is m-invariant in* $\mathcal{C}$ whenever for every $\mathcal{M} \in \mathcal{C}$, every $\mathcal{N} \in \mathit{succs}^{\mathcal{M}}(m)$ and every formula φ, $\mathcal{M} \models [n]\varphi$ iff $\mathcal{N} \models [n]\varphi$.

We can then prove the following result:

Proposition 11.1. *Let m and n be two modalities and let $\mathcal{C}$ be a class of models. The following two conditions are equivalent:*

1. *n is m-invariant in $\mathcal{C}$*
2. *$\mathcal{C} \models ([n]\varphi \to [m][n]\varphi) \wedge (\langle m\rangle[n] \to [n]\varphi)$, for all φ.*

Moreover, if n is m-invariant in $\mathcal{C}$ then n is $\mathcal{C}$-extractable from m.

Proof. We only prove that 1 and 2 are equivalent. The fact that whenever n is m-invariant in $\mathcal{C}$ then n is $\mathcal{C}$-extractable from m will be a corollary of Theorem 11.1.

$1 \Rightarrow 2$) Assume 1; we have to prove that both $\mathcal{M} \models [n]\varphi \to [m][n]\varphi$ and $\mathcal{M} \models \langle m\rangle[n] \to [n]\varphi$ hold. For the former, assume also $\mathcal{M} \models [n]\varphi$, in which case 1 directly implies $\mathcal{M} \models [m][n]\varphi$. For the latter, assume $\mathcal{M} \models \langle m\rangle[n]\varphi$ and let $\mathcal{N} \in \mathit{succs}^{\mathcal{M}}(m)$ be such that $\mathcal{N} \models [n]\varphi$; again 1 directly implies $\mathcal{M} \models [n]\varphi$.

$2 \Rightarrow 1$) Assume 2 and let $\mathcal{N} \in \mathit{succs}^{\mathcal{M}}(m)$. If $\mathcal{M} \models [n]\varphi$ then $\mathcal{M} \models [m][n]\varphi$ and $\mathcal{N} \models [n]\varphi$. If $\mathcal{N} \models [n]\varphi$ then $\mathcal{M} \models \langle m\rangle[n]\varphi$ and hence $\mathcal{M} \models [n]\varphi$. □

Saying that n is m-invariant in $\mathcal{C}$ is strictly stronger than saying that n is $\mathcal{C}$-extractable from m (and it is for that reason that we say that m-invariance is only a necessary condition for extractability). But the additional strength carried out by this notion results in some useful properties. In particular, invariance of a set of pairs of modalities implies its extractability.

Proposition 11.2. *Let $\mathcal{C}$ be a class of models, and let S be a set of pairs of modalities such that $(m, n) \in S$ implies that n is m-invariant in $\mathcal{C}$. Then S is $\mathcal{C}$-extractable.*

Proof. The result follows from Theorem 11.1. □

Notice that Proposition 11.2 is not true if we only require that for each pair $(m, n) \in S$, n is $\mathcal{C}$ extractable from m, as was shown in Example 11.4.

Fix a class of models $\mathcal{C}$, and let S be the set of all pairs (m, n) such that n is m-invariant in $\mathcal{C}$. Proposition 11.2 says that any formula φ is $\mathcal{C}$-equivalent to a formula φ' where no n occurs in the immediate scope of m

for $(m, n) \in S$. But this is implicitly defining a normal form for $\mathcal{C}$, one where all the modalities that are extractable (because of m-invariance) are in fact extracted. Example 11.4 shows that we cannot derive a similar normal form from general extractability.

Definition 11.5 ($\mathcal{C}$-extracted form)**.** We say that φ is in *$\mathcal{C}$-extracted normal form* (or *$\mathcal{C}$-extracted form* for short) if for every n that occurs in φ in the immediate scope of m, n is *not* m-invariant in $\mathcal{C}$.

We will see in Section 11.2 how to compute $\mathcal{C}$-extracted forms. Before that, we will show that it is possible to give (sufficient) conditions on classes of models that ensure m-invariance. That is, we can define conditions under which we may guarantee m-invariance by looking only at the accessibility relation for the involved modalities (i.e., disregarding the rest of the language). Item 2 in Proposition 11.1 gives us a hint for this purely structural characterization.

Definition 11.6 ($\mathcal{C}$-transitive and $\mathcal{C}$-euclidean pair)**.** Given modalities m and n, we say that (m, n) *is a $\mathcal{C}$-transitive pair* if for every $\mathcal{M} \in \mathcal{C}$ and every $\mathcal{N} \in \mathit{succs}^{\mathcal{M}}(m)$, $\mathit{succs}^{\mathcal{N}}(n) \subseteq \mathit{succs}^{\mathcal{M}}(n)$. Similarly, we say that (m, n) *is a $\mathcal{C}$-euclidean pair* if for every $\mathcal{M} \in \mathcal{C}$ and every $\mathcal{N} \in \mathit{succs}^{\mathcal{M}}(m)$, $\mathit{succs}^{\mathcal{M}}(n) \subseteq \mathit{succs}^{\mathcal{N}}(n)$.

If the pair (m, m) is $\mathcal{C}$-transitive we will just say that m is $\mathcal{C}$-transitive and, similarly, we will say that m is $\mathcal{C}$-euclidean if (m, m) is $\mathcal{C}$-euclidean.

Proposition 11.3. *If (m, n) is a $\mathcal{C}$-transitive and $\mathcal{C}$-euclidean pair, then n is m-invariant in $\mathcal{C}$ (and, hence, $\mathcal{C}$-extractable from m).*

Proof. Suppose $\mathcal{M} \in \mathcal{C}$ and $\mathcal{N} \in \mathit{succs}^{\mathcal{M}}(m)$. Since m and n form a $\mathcal{C}$-transitive and a $\mathcal{C}$-euclidean pair we know $\mathit{succs}^{\mathcal{N}}(n) = \mathit{succs}^{\mathcal{M}}(n)$, from which it follows that $\mathcal{M} \models [n]\varphi$ iff $\mathcal{N} \models [n]\varphi$. □

There are cases where even simpler conditions can be given.

Definition 11.7 ($\mathcal{C}$-relational and $\mathcal{C}$-constant)**.** We say that a modality m is *$\mathcal{C}$-relational* whenever $\mathcal{C} \subseteq \mathcal{C}^K_m$, and we say that m is *$\mathcal{C}$-constant* if $R^{\mathcal{M}}(m, u) = R^{\mathcal{M}}(m, v)$, for all $\mathcal{M} \in \mathcal{C}$ and all $u, v \in |\mathcal{M}|$.

Example 11.5. As an example of a modality that is constant but not relational, we can consider the reset_i modality (for some $i \in \mathsf{Nom}$) with respect to the class $\mathcal{C}_{\mathsf{reset}_i} \subset \mathcal{C}_{\mathsf{Nom}}$ given by the following defining condition:

$$\begin{aligned} P_{\mathsf{reset}_i}(\mathcal{M}) \iff R^{\mathcal{M}}(\mathsf{reset}_i, x) &= \{\langle w_0, |\mathcal{M}|, V, R^{\mathcal{M}} \rangle\}, \text{ where} \\ V(x) &= \begin{cases} w_0 & \text{if } x \in \mathsf{Nom} \\ V^{\mathcal{M}}(x) & \text{otherwise} \end{cases} \\ \{w_0\} &= V^{\mathcal{M}}(i). \end{aligned}$$

It is easy to see that reset_i is $\mathcal{C}_{\mathsf{reset}_i}$-constant: the reset_i-successor of $\mathcal{M}$ depends exclusively on $V^{\mathcal{M}}$. And since the valuation of this successor may be different from that of $\mathcal{M}$ it is clear that it is not $\mathcal{C}_{\mathsf{reset}_i}$-relational.

Observe that if $\mathcal{C}' \subseteq \mathcal{C}$ and m is $\mathcal{C}$-relational ($\mathcal{C}$-constant) then m is also $\mathcal{C}'$-relational ($\mathcal{C}$-constant). It is straightforward to verify that $@_i$ is $\mathcal{C}_{@_i}$-relational and $\mathcal{C}_{@_i}$-constant (i needs not be a nominal). Similarly, A is $\mathcal{C}_{\mathsf{A}}$-relational and $\mathcal{C}_{\mathsf{A}}$-constant.

One can prove that if m is $\mathcal{C}$-relational and n is $\mathcal{C}$-constant then (m, n) is a $\mathcal{C}$-transitive and $\mathcal{C}$-euclidean pair, and hence, n is $\mathcal{C}$-extractable from m.

Proposition 11.4. *If m is $\mathcal{C}$-relational and n is $\mathcal{C}$-constant, then n is m-invariant in $\mathcal{C}$.*

As particular instances of Proposition 11.4 we obtain the following results (where we call a modality m self $\mathcal{C}$-invariant if m is m-invariant in $\mathcal{C}$).

Corollary 11.1.

1. *If $\mathcal{C} \subseteq \mathcal{C}^K_m \cap \mathcal{C}_{@_i}$, then $@_i$ is m-invariant in $\mathcal{C}$.*
2. *If $\mathcal{C} \subseteq \mathcal{C}^K_m \cap \mathcal{C}_{\mathsf{A}}$, then A is m-invariant in $\mathcal{C}$.*
3. *If $m \in \{\mathsf{A}, @_i, \downarrow i, ⓔ, ⓡ, ⓕ\}$ then m is self $\mathcal{C}_m$-invariant.*

But Proposition 11.4 is strictly weaker than Proposition 11.3. As an example, it is easy to use Proposition 11.3 to show that ⓔ is $\mathcal{C}_{ⓔ} \cap \mathcal{C}_{ⓡ}$-extractable from ⓡ and $\mathcal{C}_{ⓔ} \cap \mathcal{C}_{ⓕ}$-extractable from ⓕ. None of these modalities is relational.

11.2 Computing Formulas in Extracted Form

In this section we will show that for every class $\mathcal{C}$ there exists a computable function $f_{\mathcal{C}}$ such that $f_{\mathcal{C}}(\varphi)$ is in $\mathcal{C}$-extracted form (cf. Definition 11.5) and $\mathcal{C} \models \varphi \leftrightarrow f_{\mathcal{C}}(\varphi)$, for all φ. Throughout this section we will make use of formulas in modal conjunctive normal form ($\mathrm{CNF}_\Box$).

Definition 11.8 ($\mathrm{CNF}_\Box$). We say that a formula is a $\mathrm{CNF}_\Box$-*literal* when it is of the form a, $\neg a$, $[m]\psi$ or $\neg[m]\psi$, for $a \in \mathsf{Atom}$ and ψ a $\mathrm{CNF}_\Box$-*clause*. We say φ is a $\mathrm{CNF}_\Box$-*clause* whenever φ is a $\mathrm{CNF}_\Box$-literal or is of the form $\psi_1 \vee \psi_2$ with ψ_1 and ψ_2 $\mathrm{CNF}_\Box$-clauses. Finally a $\mathrm{CNF}_\Box$-*formula* is a "conjunction" $\neg(\bigvee_i \neg\psi_i)$ of $\mathrm{CNF}_\Box$-clauses ψ_i.

It follows from this definition that a $\mathrm{CNF}_\Box$-formula φ cannot contain two consecutive negations (i.e., $\neg\neg\psi$ is not a subformula of φ). Moreover, if $[m]\psi$ occurs in a $\mathrm{CNF}_\Box$-formula, then ψ must be either a (negated) atom, a disjunction where no disjunct is a conjunction, or a (negated) formula of the form $[n]\chi$ (with m and n not necessarily different). We use this observation to show that it is simple to obtain a $\mathcal{C}$-extracted formula from one in $\mathrm{CNF}_\Box$.

Lemma 11.1. *For every class $\mathcal{C}$, there exists a computable function $f_{\mathcal{C}}$ such that for all φ in* $\mathrm{CNF}_{\Box}$*, $f_{\mathcal{C}}(\varphi)$ is a* $\mathrm{CNF}_{\Box}$*-formula in $\mathcal{C}$-extracted form and $\mathcal{C} \models \varphi \leftrightarrow f_{\mathcal{C}}(\varphi)$.*

Proof. Given a class $\mathcal{C}$, let $E_{\mathcal{C}}$ be the rewrite system that contains, for every n that is m-invariant in $\mathcal{C}$, the following rules (modulo commutativity of $\vee$):

$$[m][n]\varphi \longrightarrow_{E_{\mathcal{C}}} [m]\bot \vee [n]\varphi \tag{11.3}$$

$$[m]\neg[n]\varphi \longrightarrow_{E_{\mathcal{C}}} [m]\bot \vee \neg[n]\varphi \tag{11.4}$$

$$[m](\psi \vee [n]\varphi) \longrightarrow_{E_{\mathcal{C}}} [m]\bot \vee ([m]\psi \vee [n]\varphi) \tag{11.5}$$

$$[m](\psi \vee \neg[n]\varphi) \longrightarrow_{E_{\mathcal{C}}} [m]\bot \vee ([m]\psi \vee \neg[n]\varphi) \tag{11.6}$$

It is not hard to see that $E_{\mathcal{C}}$ is terminating (but not confluent, although this is not relevant here). From the observations above, it is also straightforward to see that if a $\mathrm{CNF}_{\Box}$-formula φ is not in $\mathcal{C}$-extracted form, then there is an occurrence of m and n that satisfies the left-hand-side of one of the rules. Moreover, since the rules preserve clausal form, we can conclude that every $\mathrm{CNF}_{\Box}$-formula is rewritten to a $\mathrm{CNF}_{\Box}$-formula in $\mathcal{C}$-extracted form.

It only remains to see that the resulting formula is $\mathcal{C}$-equivalent; for this, it suffices to show that both sides of each rule are actually $\mathcal{C}$-equivalent. We will discuss $\mathcal{C} \models [m](\psi \vee \neg[n]\varphi) \leftrightarrow [m]\bot \vee ([m]\psi \vee \neg[n]\varphi)$, the other cases are analogous. Suppose, then, that for $\mathcal{M} \in \mathcal{C}$, $\mathcal{M} \models [m](\psi \vee \neg[n]\varphi)$. It could be the case that $\mathcal{M}$ has no m-successors, or that every m-successor satisfies ψ, but then, clearly, $\mathcal{M} \models [m]\bot \vee [m]\psi$. Suppose, then that for some $\mathcal{N} \in \mathit{succs}^{\mathcal{M}}(m)$, $\mathcal{N} \models \neg\psi \wedge \neg[n]\varphi$. Since n is m-invariant in $\mathcal{C}$, we know $\mathcal{M} \models [n]\varphi$ iff $\mathcal{N} \models [n]\varphi$, so we can conclude that $\mathcal{M} \models \neg[n]\varphi$. The other direction is proved in a similar way. □

We can now prove the main result of this section. Observe that both Proposition 11.2 and the bit of Proposition 11.1 that was left unproven will follow as a trivial corollary.

Theorem 11.1. *For every class $\mathcal{C}$, there exists a translation $f_{\mathcal{C}}$ such that for all φ, $f_{\mathcal{C}}(\varphi)$ is in $\mathcal{C}$-extracted form and $\mathcal{C} \models \varphi \leftrightarrow f_{\mathcal{C}}(\varphi)$.*

Proof. From Lemma 11.1, this proof simply amounts to observing that every formula can be recursively turned to an equivalent one in $\mathrm{CNF}_{\Box}$, which is a folklore result. For the sake of completeness, we include such a computable transformation. Let $\longrightarrow_{\mathrm{CNF}_{\Box}}$ be the rewrite system that contains the following rules (modulo commutativity):

$$\neg\neg\varphi \longrightarrow_{\mathrm{CNF}_{\Box}} \varphi \tag{11.7}$$

$$\varphi \vee \neg(\psi \vee \chi) \longrightarrow_{\mathrm{CNF}_{\Box}} \neg(\neg(\varphi \vee \neg\psi) \vee \neg(\varphi \vee \neg\chi)) \tag{11.8}$$

$$[m]\neg(\varphi \vee \psi) \longrightarrow_{\mathrm{CNF}_{\Box}} \neg(\neg[m]\neg\varphi \vee \neg[m]\neg\psi) \tag{11.9}$$

The last rule, of course, must be instantiated for every $m \in \mathsf{Mod}$. It is easy to see that the left and right-hand-sides of each rule are logically equivalent. Rule (11.7) eliminates double negations and rules (11.8) and (11.9) are simply distribution of disjunction and box, respectively, over conjunctions. If a formula is not in CNF$_\Box$, then it may be rewritten using one of these rules; and since the system can be shown to be terminating, it constitutes the desired computable transformation. □

Of course, since the proof above uses a translation to CNF$_\Box$, the formula in extracted form may end up being exponentially larger than the original one. This begs the question if there exists an alternative translation with only a polynomial overhead. The answer is negative for the general case, as is witnessed by the following result.

Theorem 11.2 (ten Cate [2005])**.** *There is no polynomial translation from $\mathcal{H}(@)$-formulas to $\mathcal{C}_{\mathcal{H}(@)}$-equivalent formulas in $\mathcal{C}_{\mathcal{H}(@)}$-extracted form.*

On the other hand, if we restrict ourselves to satisfiability preserving translations, then it is indeed possible to give polynomial translations for arbitrary classes.

Theorem 11.3. *For every class $\mathcal{C}$, there exists a polynomial translation $f_{\mathcal{C}}$ such that for all φ, $f_{\mathcal{C}}(\varphi)$ is in $\mathcal{C}$-extracted form, and φ is satisfiable iff $f_{\mathcal{C}}(\varphi)$ is satisfiable.*

Proof. It is easy to verify that the transformation used in the proof of Lemma 11.1 runs in polynomial time. Hence, we only need to provide a polynomial, satisfiability preserving translation to CNF$_\Box$. Mints [1989] exhibits a sequent-based polynomial translation for the case of the basic unimodal logic. We will use instead a rewriting based translation similar to the ones above. Observe that in the proof of Theorem 11.1, the source for the exponential blow-up is in rule (11.8) where φ occurs twice on its right-hand-side. The rewrite system $\longrightarrow_{\text{p-CNF}_\Box}$ is obtained by replacing this rule by one that uses additional proposition symbols:

$$\neg\neg\varphi \longrightarrow_{\text{p-CNF}_\Box} \varphi \tag{11.10}$$
$$\varphi \vee \neg(\psi \vee \chi) \longrightarrow_{\text{p-CNF}_\Box} \neg(\neg(\varphi \vee \neg p_\varphi) \vee \neg(p_\varphi \vee \psi) \vee \neg(\neg p_\varphi \vee \chi)) \tag{11.11}$$
$$[m]\neg(\varphi \vee \psi) \longrightarrow_{\text{p-CNF}_\Box} \neg(\neg[m]\neg\varphi \vee \neg[m]\neg\psi) \tag{11.12}$$

It is routine to see that $\longrightarrow_{\text{p-CNF}_\Box}$ is terminating (but not confluent, in fact different reduction strategies may introduce different proposition symbols). In any case, it is not hard to see that the length of every derivation is bound by a polynomial. Finally, one needs to show that every rewrite step leads to an equi-satisfiable formula, but this is also straightforward. □

11.3 Modal depth and formula complexity

The modal depth of a formula is often taken as a measure of its complexity. Its appeal lies in that it estimates and summarizes in a single value several aspects of complexity: the expressive power of the formula, the computational cost of evaluating the formula in a model, the minimum size of a model for the formula, etc.

Now, suppose $\mathcal{C} \models \varphi \leftrightarrow \varphi'$ and $md(\varphi) > md(\varphi')$. It would often make sense to prefer $md(\varphi')$ as a more accurate estimation of the complexity of φ, especially if the cost of computing $md(\varphi')$ can be disregarded. Theorem 11.1 tells us that $\mathcal{C} \models \varphi \leftrightarrow f_{\mathcal{C}}(\varphi)$ and $md(\varphi) \geq md(f_{\mathcal{C}}(\varphi))$. But, while $md(f_{\mathcal{C}}(\varphi))$ appears to be a more accurate measure of the complexity of φ than $md(\varphi)$, there are two drawbacks. First, $f_{\mathcal{C}}$ is not univoquely defined in Theorem 11.1 (since different rewrite strategies lead to different normal forms). Second, even if we fix a definition for $f_{\mathcal{C}}$ (e.g., by fixing a rewrite strategy) we already saw that $f_{\mathcal{C}}(\varphi)$ can be exponentially larger than φ and hence, first computing $f_{\mathcal{C}}(\varphi)$ and then obtaining its modal depth would be too expensive.

In this section, by using a generalization of the notion of modal depth (cf. the definition of "modal paths" below) we will be able to give more accurate invariance results. These results will cover a wide spectrum of complexity measures computable in polynomial time.

As a generalization of modal depth we will take all the sequences of modalities occurring in some branch of the formation tree of a formula. We will call this set the *modal paths* of a formula.

Definition 11.9 (Modal paths)**.** We define $\pi(\varphi) \subset \mathsf{Mod}^*$, the set of *modal paths* of a formula φ, in an inductive way:

$$\begin{array}{rcl} \pi(a) & = & \{\epsilon\} \\ \pi(\neg\psi) & = & \pi(\psi) \\ \pi(\psi \vee \chi) & = & \pi(\psi) \cup \pi(\chi) \\ \pi([m]\psi) & = & \{m.p \mid p \in \pi(\psi)\} \end{array}$$

Observe that $md(\varphi) = \max\{|t| \mid t \in \pi(\varphi)\}$, where $|t|$ is the length of t and that $\pi(\varphi)$ can be computed in polynomial time. We will use π to define complexity measures which are more accurate than md.

Definition 11.10 (Extracted variants and k-bounds)**.** Given two finite sets $M, M' \subset \mathsf{Mod}^*$, we say that M' *is a $\mathcal{C}$-extracted variant of* M, and notated $M' \in \tau_{\mathcal{C}}(M)$, if M' can be obtained from M by repeatedly applying the following rule until it can no longer be applied (here, $s, t \in \mathsf{Mod}^*$ while $m, n \in \mathsf{Mod}$):

$$\frac{A \cup \{s.m.n.t\}}{A \cup \{s.m, s.n.t\}}\ n \text{ is } m\text{-invariant in } \mathcal{C} \tag{11.13}$$

Finally, we will say that a formula φ is *k-bounded in $\mathcal{C}$* whenever there exists $M \in \tau_{\mathcal{C}}(\pi(\varphi))$ such that $k \geq \max\{|t| \mid t \in M\}$.

Any function c will be a correct measure of complexity for $\mathcal{C}$ as long as φ is $c(\varphi)$-bounded in $\mathcal{C}$ for all φ. Trivially, then, md will be correct for all $\mathcal{C}$. The following theorem links k-bounds with k-bisimulations, justifying this claim.

Theorem 11.4. *Let $\mathcal{M}, \mathcal{N} \in \mathcal{C}$ be such that $\mathcal{M} \underline{\leftrightarrow}_k \mathcal{N}$. Then, for all φ k-bounded in $\mathcal{C}$, $\mathcal{M} \models \varphi$ iff $\mathcal{N} \models \varphi$.*

Proof. First, observe that, for all ψ in CNF$_\Box$, if M is obtained from $\pi(\psi)$ by applying rule (11.13) once, then there exists a derivation $\psi \longrightarrow_{E_\mathcal{C}} \psi_1 \longrightarrow_{E_\mathcal{C}} \cdots \longrightarrow_{E_\mathcal{C}} \psi_n$ such that $\pi(\psi_n) = M'$ (notice that if $n > 1$ then there exist more than one path in the syntactic tree of ψ where the rewritten modal path occurs). Conversely, if $M = \pi(\psi)$ and rule (11.13) cannot be applied to M, then ψ must be in extracted form.

Now, since φ is k-bounded, there must be a derivation of M from $\pi(\varphi)$ such that $k \geq \max\{|t| \mid t \in M\}$. Let φ' be a CNF$_\Box$-formula $\mathcal{C}$-equivalent to φ obtained using the rewrite system $\longrightarrow_{\mathrm{CNF}_\Box}$ (cf. Theorem 11.1). It is trivial to see that $\pi(\varphi) = \pi(\varphi')$, but this means that we can use the observation above to derive a $\mathcal{C}$-equivalent φ'' from φ' using $\longrightarrow_{E_\mathcal{C}}$ such that $\pi(\varphi'') = M$ and φ'' is in $\mathcal{C}$-extracted form. Therefore we have $md(\varphi'') \leq k$ and using Theorem 10.2, $\mathcal{M} \models \varphi''$ iff $\mathcal{N} \models \varphi''$. Finally, since φ, φ' and φ'' are all $\mathcal{C}$-equivalent, we conclude $\mathcal{M} \models \varphi$ iff $\mathcal{N} \models \varphi$. $\Box$

To illustrate this result, we propose next a (polynomially computable) complexity measure for $\mathcal{H}(@)$ and show it is correct and more accurate than *md*.

Definition 11.11 (Hybrid depth)**.** We define $\mathsf{hd}(\varphi)$, the *hybrid depth* of a $\mathcal{H}(@)$-formula φ as:

$$\mathsf{hd}(\varphi) \overset{def}{=} \max\left\{\mathsf{hd}'(\varphi), 1 + \max\{\mathsf{hd}'(\psi) \mid [@_i]\psi \text{ is a subformula of } \varphi\}\right\},$$

where:

$$\begin{array}{lcl}
\mathsf{hd}'(a) & = & 0,\ \text{for } a \in \mathsf{Atom} \\
\mathsf{hd}'(\neg\varphi) & = & \mathsf{hd}'(\varphi) \\
\mathsf{hd}'(\varphi \vee \psi) & = & \max\{\mathsf{hd}'(\varphi), \mathsf{hd}'(\psi)\} \\
\mathsf{hd}'([@_i]\varphi) & = & 0 \\
\mathsf{hd}'([r]\varphi) & = & 1 + \mathsf{hd}'(\varphi),\ \text{for } r \in \mathsf{Rel}.
\end{array}$$

Clearly, hd is computable in polynomial time and $\mathsf{hd}(\varphi) \leq md(\varphi)$, for all φ. Moreover, for every k there exists φ_k such that $md(\varphi_k) = \mathsf{hd}(\varphi_k) + k$. In other words, when considering the set of all hybrid formulas, hd can be found to be arbitrarily smaller than md. Still, hd is a correct measure of complexity for $\mathcal{H}(@)$.

Proposition 11.5. *For every $\mathcal{H}(@)$-formula φ, φ is $\mathsf{hd}(\varphi)$-bound in $\mathcal{C}_{\mathcal{H}(@)}$.*

Proof. Observe that rule (11.13) is confluent in the case of $\mathcal{C}_{\mathcal{H}(@)}$ (it need not be in the general case) so there is only one $M \in \tau_{\mathcal{C}}(\pi(\varphi))$. Moreover, if $s \in M$, then either $s \in \mathsf{Rel}^*$ or $s = @_i.t$ with $t \in \mathsf{Rel}^*$. In the last case it must be because $@_i\psi$ with $t \in \pi(\psi)$ occurs in φ. In any case, it is clear that $\mathsf{hd}(\varphi) = \max\{|s| \mid s \in M\}$. $\square$

As a final application of Theorem 11.4 we exhibit an alternative proof for a classic result in computational complexity for modal logics.

Theorem 11.5 (Ladner [1977])**.** *The satisfiability problem for* S5 *is NP-complete.*

Proof. Recall that S5 is obtained by adding axioms T, 4 and 5 (cf. Section 1.2.3) to the basic modal logic K. The lower bound is given by the complexity of the satisfiability problem for propositional logic. For the upper bound, assume a signature $\mathcal{S} = \langle \mathsf{Atom}, \{m\}\rangle$ and take $\mathcal{C}_{\mathbf{S5}}$ to be the class of all $\mathcal{S}$-models where m is interpreted as a relational, transitive, reflexive and symmetric modality. Since this class satisfies axiom 5, we know m has to be euclidean too. Suppose we are looking a model for φ. By Proposition 11.3, m is self $\mathcal{C}$-extractable. It is easy to see that then if $M \in \tau_{\mathcal{C}}(\pi(\varphi))$ then M is the singleton set $\{m\}$. Therefore, φ is 1-bounded in $\mathcal{C}_{S5}$ and, therefore it suffices to non-deterministically guess a proper model $\mathcal{M}$ of depth 1 with enough successors (e.g. the number of occurrences of $[m]$ in φ) and verify if $\mathcal{M} \models \varphi$, which can be done in polynomial time. $\square$

Bibliography

H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998. 5.1

C. Areces. *Logic Engineering. The Case of Description and Hybrid Logics*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands, October 2000. 1.1, 1.2.2

C. Areces and D. Gorín. Resolution with order and selection for hybrid logics. Submitted to the Journal of Automated Reasoning, 2009. 7, 8.1, 8.4

C. Areces and B. ten Cate. Hybrid logics. In Blackburn et al. [2006], chapter 14, pages 821–868. 1.2.1, 10

C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in LNCS, pages 307–321. Springer, 1999. Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 1999. 1.1, 1.2.1

C. Areces, R. Gennari, J. Heguiabere, and M. de Rijke. Tree-based heuristics in modal theorem proving. In W. Horn, editor, *Proc. of ECAI'2000*, pages 199–203, Berlin, Germany, 2000. 3.1, 5.2, 5.2, 5.1, 5.2

C. Areces, P. Blackburn, and M. Marx. Hybrid logics: Characterization, interpolation and complexity. *Journal of Symbolic Logic*, 66(3):977–1010, 2001a. 1.2.2

C. Areces, H. de Nivelle, and M. de Rijke. Resolution in modal, description and hybrid logic. *Journal of Logic and Computation*, 11(5):717–736, 2001b. 2.2, 2.2, 3.2

C. Areces, D. Figueira, S. Figueira, and S. Mera. Expressive power and decidability for memory logics. In *Logic, Language, Information and Computation*, volume 5110 of *Lecture Notes in Computer Science*, pages 56–68.

Springer Berlin / Heidelberg, 2008. Proceedings of WoLLIC 2008. 3.3, 10.1

Y. Auffray and P. Enjalbert. Modal theorem proving: An equational viewpoint. In N. S. Sridharan, editor, *Proc. of the 11th International Joint Conference on Artificial Intelligence*, pages 441–445. Morgan Kaufmann Pub., 1989. 6

Y. Auffray and P. Enjalbert. Modal theorem proving: An equational viewpoint. *Journal of Logic and Computation*, 2(3):247–295, 1992. 6, 6.3

Y. Auffray, P. Enjalbert, and J. Hebrard. Strategies for modal resolution: results and problems. *Journal of Automated Reasoning*, 6(1):1–38, 1990. 2.2

F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, New York, NY, USA, 1998. 1, 7.2, 7.2, 7.2

L. Bachmair and H. Ganzinger. Resolution theorem proving. In Robinson and Voronkov [2001], chapter 2, pages 19–99. 2.1, 2.1, 2.1, 2.1, 9, 9.2

L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4 (3):217–247, 1994. 2.1

H. Barendregt. *The lambda calculus: its syntax and semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, rev. ed edition, 1984. 1.2.1

P. Blackburn. Internalizing labelled deduction. *Journal of Logic and Computation*, 10(1):137–168, 2000. 3

P. Blackburn and M. Marx. Tableaux for quantified hybrid logic. In U. Egly and C. Fermüller, editors, *Automated Reasoning with Analytic Tableaux and Related Method*, number 2381 in LNAI, pages 38–52. Springer Verlag, 2002. 3

P. Blackburn and B. ten Cate. Pure extensions, proof rules, and hybrid axiomatics. *Studia Logica*, 84(2):277–322, 11 2006. 4.2, 4.2, 1, 4.2, 4.2

P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Sciene*. Cambridge University Press, 2002. 1.2.3, 5.2, 5.3, 10.1

P. Blackburn, F. Wolter, and J. van Benthem, editors. *Handbook of Modal Logics*, volume 3 of *Studies in Logic and Practical Reasoning)*. Elsevier, 2006. 1, 1.1, 11.3

T. Bolander and P. Blackburn. Termination for hybrid tableaus. *Journal of Logic and Computation*, 17(3):517–554, 2007. 3

T. Bolander and P. Blackburn. Terminating tableau calculi for hybrid logics extending *K*. *Electron. Notes Theor. Comput. Sci.*, 231:21–39, 2009. 3

T. Bolander and T. Brauner. Tableau-based decision procedures for hybrid logic. *Journal of Logic and Computation*, 16(6):737–763, 2006. 3

D. Brand. Proving theorems with the modification method. *SIAM Journal on Computing*, 4(4):412–430, 1975. ii., 2.1

T. Braüner. Natural deduction for hybrid logic. *Journal of Logic and Computation*, 14(3):329–353, 2004a. 3

T. Braüner. Two natural deduction systems for hybrid logic: A comparison. *Journal of Logic, Language and Information*, 13(1):1–23, 2004b. 3

E. Clarke, O. Grumberg, and D. Peled. *Model checking.* The MIT Press, Cambridge, Massachusetts, 1999. 1.1

H. de Nivelle and M. de Rijke. Deciding the guarded fragments by resolution. *Journal of Symbolic Computation*, 35(1):21–58, January 2003. 5.1

H. de Nivelle and I. Pratt-Hartmann. A resolution-based decision procedure for the two-variable fragment with equality. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of IJCAR 2001*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 211–225. Springer-Verlag, 2001. 5.1

P. Enjalbert and L. Fariñas del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65(1):1–33, 1989. 2.2

L. Fariñas del Cerro. A simple deduction method for modal logic. *Information Processing Letters*, 14(2):49–51, 1982. 2.2

L. Fariñas del Cerro and A. Herzig. Linear modal deductions. In E. L. Lusk and R. A. Overbeek, editors, *Proc. of the 9th Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 487–499. Springer, 1988. 6

Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2): 194–211, 1979. 1.1

M. Fitting. Destructive modal resolution. *Journal of Logic and Computation*, 1(1):83–97, 1990. 2.2

D. Gabbay. Expressive functional completeness in tense logic (preliminary report). In U. Mönnich, editor, *Aspects of Philosophical Logic*, volume 147 of *Synthese Library*, pages 91–117. Reidel, 1981. 5.1

H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In Giuseppe Longo, editor, *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science (LICS-99)*, pages 295–303, Trento, Italy, 1999. IEEE Computer Society, IEEE Computer Society. 5.1

M. Giese and W. Ahrendt. Hilbert's ϵ-terms in automated theorem proving. In N. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, Intl. Conf. (TABLEAUX'99)*, volume 1617 of *LNAI*, pages 171–185. Springer, 1999. 3.2, 8.3

R. Goldblatt and S. Thomason. Axiomatic classes in propositional modal logic. In *Algebra and Logic*, volume 450 of *Lecture Notes in Mathematics*, pages 163–173. Springer Berlin / Heidelberg, 1975. 1.2.3, 6.4

E. Grädel. Why are modal logics so robustly decidable? In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science. Entering the 21st Century*, pages 393–408. World Scientific, 2001. 5.2

D. Harel, D. Kozen, and J. Tiuryn. *Dynamic logic*. Foundations of Computing. MIT Press, Cambridge, Mass., 2000. 1.1

M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985. 1.2.2

J. Herbrand. *Recherches sur la théorie de la démonstrations.* PhD thesis, Sorbone, Paris, 1930. Reprinted in W. Goldfarb, editor, *Logical Writings.* Reidel, 1971. 2.1

D. Hilbert and P. Bernays. *Grundlagen der Mathematik*, volume 2. Springer, 1939. 8.3

I. Horrocks, U. Hustadt, U. Sattler, and R. Schmidt. Computational modal logic. In Blackburn et al. [2006], pages 181–245. 2

Ian Horrocks and Ulrike Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 39(3):249–276, 2007. 3

U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, 9(4):479–522, 1999. 6.4

B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:222–259, 1997. 3.3

W. Joyner, Jr. Resolution strategies as decision procedures. *Journal of the ACM*, 23(3):398–417, 1976. 6.3, 8.2

Mark Kaminski and Gert Smolka. Terminating tableau systems for hybrid logic with difference and converse. *Journal of Logic, Language and Information*, 18(4):437–464, Oct 2009. 3

Yevgeny Kazakov and Boris Motik. A resolution-based decision procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 40(2–3):89–116, 2008. 3

D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Algebra*, pages 263–297. Pergamon Press, 1970. 7.2

R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977. 1.1, 11.5

A. Leisenring. *Mathematical logic and Hilbert's ϵ-symbol.* MacDonald, London, 1969. 8.3

C. Lewis. *A Survey of Symbolic Logic.* Univ. of California Press (Berkeley), Berkeley, 1918. Reprint of Chapters I–IV by Dover Publications, 1960, New York. 1

Ewing Lusk. Controlling redundancy in large search spaces: Argonne-style theorem proving through the years. In A. Voronkov, editor, *Proc. of the International Conference on Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 96–106. Springer Berlin / Heidelberg, 1992. 9.1

W. McCune and L. Wos. Experiments in automated deduction with condensed detachment. In *CADE-11: Proceedings of the 11th International Conference on Automated Deduction*, pages 209–223, London, UK, 1992. Springer-Verlag. 3.1, 4.1

B. Meyer. Applying "design by contract". *IEEE Computer*, 25(10):40–51, October 1992. 1.1

G. Mints. Gentzen-type systems and resolution rules, Part 1: Propositional logic. In *Proceedings of COLOG-88*, volume 417 of *Lecture Notes in Computer Science*, pages 198–231. Springer, 1990. 2.2

G. Mints. Resolution calculi for modal logics. In *Eight Papers Translated from the Russian*, volume 143 of *American Mathematical Society Translations*, pages 1–14. AMS, 1989. 11.2

M. Mortimer. On languages with two variables. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21(1), 1975. 3.1, 5.1

M. Mundhenk, T. Schneider, T. Schwentick, and V. Weber. Complexity of hybrid logics over transitive frames. In H. Schlingloff, editor, *Proc. of Methods for Modalities 4*, volume 194 of *Informatik-Berichte*, pages 62–78. Humboldt-Universität zu Berlin, 2005. 1.2.3

R. Nieuwenhuis and A. Rubio. Basic superposition is complete. In *Proc. of the 4th European Symposium on Programming*, volume 582 of *Lecture Notes in Computer Science*, pages 371–389. Springer-Verlag, 1992. ISBN 3-540-55253-7. 9

R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In Robinson and Voronkov [2001], chapter 7, pages 371–443. 2.1, 7.4

A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In Robinson and Voronkov [2001], chapter 6, pages 335–367. 2.1, 3.3

H. Ohlbach. A resolution calculus for modal logics. In E. L. Lusk and R. A. Overbeek, editors, *Proc. of the 9th Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 500–516. Springer, 1988a. 6

H. Ohlbach. *A Resolution Calculus for Modal Logics.* PhD thesis, Universität Kaiserslautern, 1988b. 6, 6.3

H. Ohlbach and R. Schmidt. Functional translation and second-order frame properties of modal logics. *Journal of Logic and Computation*, 7(5):581–603, 1997. 6.3, 6.3, 6.4, 6.3, 6.3

H. Ohlbach, A. Nonnegart, M. de Rijke, and D. Gabbay. Encoding two-valued non-classical logics in classical logic. In Robinson and Voronkov [2001], chapter 21, pages 1403–1486. 5.1

V. R. Pratt. Models of program logics. In *Proc. 20th IEEE Symp. Foundations of Computer Science*, pages 115–122, Oct. 1979. 1.1

F. Rabe, P. Pudlák, G. Sutcliffe, and W. Shen. Solving the $100 modal logic challenge. *Journal of Applied Logic*, 7(1):113 – 130, 2009. Special Issue: Empirically Successful Computerized Reasoning. 4.1, 4.1

G. Robinson and L. Wos. Paramodulation and theorem-proving in first-order. *Machine Intelligence*, 4:135–150, 1969. 2.1

J. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965. 2.1, 2.1

J. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning.* Elsevier and MIT Press, 2001. 2, 11.3

J. Rosser. *Logic for mathematicians.* McGraw-Hill, 1953. 4.1

D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems*, 31(4), 2009. 1.2.2

R. Schmidt. Decidability by resolution for propositional modal logics. *Journal of Automated Reasoning*, 22(4):379–396, 1999. 6.3, 6.3, 8.2

D. Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27(4):477, 1962. Contribution to the "Twenty-Eighth Annual Meeting of the Association for Symbolic Logic". 5.1

R. Sebastiani and M. Vescovi. Automated reasoning in modal and description logics via SAT encoding: the case study of $K_m/\mathcal{ALC}$-satisfiability. *Journal of Artificial Intelligence Research*, 35:343–389, 2009. 1

R. Sekar, I.V. Ramakrishnan, and A. Voronkov. Term indexing. In Robinson and Voronkov [2001], chapter 26, pages 1853–1964. 9

J. Seligman. A cut-free sequent calculus for elementary situated reasoning. Research Paper HCRC-RP, Human Communication Research Center, University of Edimburgh, 1991. 3

J. Seligman. Internalization: The case of hybrid logics. *Journal of Logic and Computation*, 11(5):671–689, 2001. 3

J. Seligman. The logic of correct description. In M. de Rijke, editor, *Advances in Intensional Logic*, Applied Logic Series, pages 107–135. Kluwer, 1997. 3

B. ten Cate. *Model theory for extended modal languages*. PhD thesis, University of Amsterdam, 2005. ILLC Dissertation Series DS-2005-01. 1.2.3, 11.2

M. Tzakova. Tableau calculi for hybrid logics. In N. Murray, editor, *Proceedings of the Conference on Tableaux Calculi and Related Methods (TABLEAUX)*, volume 1617 of *Lecture Notes in Artificial Intelligence*, pages 278–292, Saratoga Springs, USA, 1999. Springer Verlag. 3

J. van Benthem. *Modal Correspondence Theory*. PhD thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976. 1.2.2, 1.2.2

M. Vardi. Automata-theoretic techniques for temporal reasoning. In Blackburn et al. [2006], chapter 17, pages 971–990. 1.1

M. Vardi. Why is modal logic so robustly decidable? In N. Immerman and P. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184. American Mathematical Society, 1996. 5.2

A. Voronkov. Algorithms, datastructures, and other issues in efficient automated deduction. In *Proceedings of IJCAR 2001*, number 2083 in LNAI, pages 13–28, Siena, 2001. Springer-Verlag. 8.2, 9

C. Walther. *A many-sorted calculus based on resolution and paramodulation.* Research Notes in Artificial Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. 4.2

C. Walther. Many-sorted inferences in automated theorem proving. In *Sorts and Types in Artificial Intelligence*, volume 418 of *Lecture Notes in Computer Science*, pages 18–48. Springer Berlin / Heidelberg, 1989. 6.4

C. Weidenbach. Combining superposition, sorts and splitting. In Robinson and Voronkov [2001], chapter 27, pages 1965–2014. 6.4

C. Weidenbach, R. Schmidt, T. Hillenbrand, R. Rusev, and D. Topic. System description: Spass version 3.0. In F. Pfenning, editor, *Proc. of CADE-21*, number 4603 in Lecture Notes in Artificial Intelligence, pages 514–520. Springer-Verlag, 2007. 6.4

N. Zamov. Modal resolutions. *Soviet Math*, 33(9):23–29, 1989. 6

Index