Università degli Studi di Torino

Dottorato di Ricerca in Informatica
(*IX ciclo*)

PhD Thesis

(Revised version dated July 9, 2003)

Normal Multimodal Logics: Automatic Deduction and Logic Programming Extension

Matteo Baldoni

Advisor:	Prof.	Alberto Martelli
Co-Advisor:	Dr.	Laura Giordano

Dipartimento di Informatica — Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino (Italy)
`http://www.di.unito.it/`

Abstract

In this thesis we work on *normal multimodal logics*, that are general modal systems with an arbitrary set of normal modal operators, focusing on the class of *inclusion modal logics.* This class of logics, first introduced by Fariñas del Cerro and Penttonen, includes some well-known *non-homogeneous* multimodal systems characterized by *interaction* axioms of the form $[t_1][t_2]\dots[t_n]\varphi \supset [s_1][s_2]\dots[s_m]\varphi$, that we call inclusion axioms.

The thesis is organized in two part. In the first part the class of inclusion modal logics is deeply studied by introducing the the *syntax*, the *possible-worlds semantics*, and the *axiomatization.* Afterwards, we define a proof theory based on an *analytic tableau calculus.* The main feature of the calculus is that it can deal in a *uniform way* with any multimodal logics in the considered class. In order to achieve this goal, we use a *prefixed* tableau calculus *á la* Fitting, where, however, we explicitly represent accessibility relations between worlds by means of a graph and we use the characterizing axioms of the logic as *rewriting rules* which create new path among worlds in the counter-model construction. Some *(un)decidability* results for this class of logic are given. Moreover, the tableau method is extended in order to deal with a wide class of normal multimodal logics that includes the ones characterized by *serial*, *symmetric*, and *Euclidean* accessibility relations.

In the second part, we propose the logic programming language NemoLOG. This language extends the Horn clauses logic allowing free occurrences of universal modal operators in front of goals, in front of clauses, and in front of clause heads. The considered multimodal systems are the ones of the class of inclusion modal logics. The aim of our proposal is not only to extend logic languages in order to perform epistemic reasoning and reasoning about actions but especially to provide tools for software engineering (e.g. modularity and inheritance among classes) *retaining* a declarative interpretation of the programs. A proof theory is developed for NemoLOG and the soundness and completeness with respect to the model theory are shown by a fixed point construction.

a mio Papà

Contents

List of Figures

Preface

Modal logics have been intensively studied in the recent years [Stirling, 1992; Fitting, 1993; Hughes and Cresswell, 1996]. The reason is that while classical first-order logic can express relationships between terms representing members of a *flat domain*, modal logics are able to *structure* knowledge, represent *beliefs of agents* and deal with problems involving *distributed reasoning* [Konolige, 1986; Genesereth and Nilsson, 1987; Halpern and Moses, 1992] together with other attitudes in *agent systems* like, for instance, *goals*, *intention* and *obligation*. Furthermore, they are well suited for representing *dynamic* aspects and, in particular, to formalize reasoning about *actions* and *time* [Wooldridge and Jennings, 1995]. All these characteristics are achieved by the use of some additional connectives, called *modal operators*, which formalize in a more natural way reasoning about *knowledge*, *beliefs*, *dynamic changes*, *time*, and *actions*. For this reason the development of automated deduction methods has received a lot of attention (see, for instance, [Hughes and Cresswell, 1968; Fitting, 1983; Fitting, 1988; Enjalbert and Fariñas del Cerro, 1989; Wallen, 1990; Catach, 1991] and, more recently, [Ognjanović, 1994; Massacci, 1994; Fariñas del Cerro and Herzig, 1995; Governatori, 1995; Cunningham and Pitt, 1996; Beckert and Goré, 1997; Baldoni *et al.*, 1998a]).

On the other hand, *logic programs*, that use flat sets of Horn clauses for representing knowledge, enjoy some good properties, such as the notion of the *least Herbrand model* together with its *fixpoint characterization* and the possible use of *goal directed proof procedures*. These features make logic a real programming language with a clear and complete operational semantics with respect to its declarative semantics [Lloyd, 1984].

Modal extensions of logic programming join tools for formalizing and reasoning about temporal and epistemic knowledge with declarative features of logic programming languages. In particular, they support *"context abstraction"*, which allows to describe dynamic and context-dependent properties of certain problems in a natural and problem-oriented way [Orgun and Ma, 1994; Fisher and Owens, 1993a; Fariñas del Cerro and Penttonen, 1992]. All these desirable features are shown by some well-known proposals, such as TEMPLOG [Abadi and Manna, 1989; Baudinet, 1989], Temporal Prolog [Gabbay, 1987], MOLOG [Fariñas del Cerro, 1986; Balbiani *et al.*, 1988], TIM [Balbiani *et al.*, 1991], Modal Prolog [Sakakibara, 1986] and also by the proposals in [Akama, 1986; Debart *et al.*, 1992; Nonnengart, 1994; Giordano and Martelli, 1994; De Giacomo and Lenzerini, 1995; Baldoni *et al.*, 1997a; Baldoni *et al.*, 1997b].

In this thesis we work on *normal multimodal logics*, that are general modal systems with an arbitrary set of normal modal operators, focusing on the class of *inclusion modal logics*. The multimodal systems which belong to this class, first introduced in [Fariñas del Cerro and Penttonen, 1988], are characterized by a set of logical axioms of the form:

$$[t_1][t_2]\dots[t_n]\varphi \supset [s_1][s_2]\dots[s_m]\varphi \quad (n > 0,\ m \geq 0)$$

that are called *inclusion axioms*. We deeply study this class of modal logics and, then, we propose a multimodal extension of logic programming, that we have called NemoLOG (which stands for New modal proLOG), based on this class of logics. Finally, some conclusions and open problems are drawn at the end of the thesis.

The thesis is organized in two part. In Part One, we, first, introduce the syntax, the possible-worlds semantics, and the axiomatization of the class of inclusion modal logics. Afterwards, we define a proof theory based on an *analytic tableau calculus*. The main feature of this calculus is that it is able to deal with the whole class of logics in a modular way with respect to the set of inclusion axioms that determines the logic. It is an extension of the calculus presented in [Nerode, 1989] which, in turn, comes from the *prefixed tableaux* in [Fitting, 1983].

Prefixed tableaux make explicit the reference to accessibility relations. In particular, in our tableau method, differently than [Fitting, 1983] (where the accessibility relations are encoded in the structure of the name of the worlds), the accessibility relations are represented by means of an explicit and separate *graph* of named nodes, each of which is associated with a set of formulae (*prefixed* formulae) and choice allows any inclusion axiom to be interpreted as a "rewriting rule" into the path structure of the graph. This is at the basis of the proofs of some (un)decidability results. Despite the fact that this kind of representation works only for those multimodal systems whose frame structure is first-order axiomatizable, we think that it is more suitable to deal with multimodal logics with arbitrary interaction axiom than the one in [Fitting, 1993], as discussed in the Chapter VII. Moreover, our tableau method can easily be extended to deal with a wide class of normal multimodal logics that includes the class of inclusion modal logics and other ones characterized by *serial*, *symmetric*, and *Euclidean* accessibility relations, as shown in Chapter VI.

In Part Two, we propose the logic programming language NemoLOG. This language extends the Horn clauses logic allowing free occurrences of universal modal operators in front of goals, in front of clauses, and in front of clause heads. The considered multimodal systems are the ones of the class of inclusion modal logics and they are specified by means of a set of particular clauses that we have called *inclusion axiom clauses*.

The aim of our proposal is not only to extend logic languages in order to perform epistemic reasoning and reasoning about actions but especially to provide tools for *software engineering* retaining a *declarative* interpretation of the programs. In particular, we will show that inclusion modal logics are well suited, on one hand, to overcome the lack of *structuring facilities* aimed at enhancing the *modularity* of logic programs (a central problem in the last years [Bugliesi *et al.*, 1994]), and, on the other, to interpret some features typical of *object-oriented* paradigms in logic programming (such as *hierarchical* dependencies and *inheritance* among classes).

A *proof theory* is developed for NemoLOG and the soundness and completeness with respect to the model theory is shown by a fixed point construction. Though the construction is pretty standard, we believe that its advantage is the *modularity* of the approach, in the sense that both the completeness and soundness proofs are *modular* with respect to the underlying inclusion modal logics of the programs.

Last but not least, we show that, in the case of programs and goals of NemoLOG, we can restrict our attention to tableau proofs of a form that recalls the one of the *uniform proof* as presented in [Miller *et al.*, 1991] and, moreover, we give a method for translating programs into standard Horn clauses, so that the translated programs can be executed by any Prolog interpreter or compiler.

Acknowledgment

I would like to thank my advisors, prof. *Alberto Martelli* and dr. *Laura Giordano*, for the help and support shown in all these years, the whole *Logic Programming and Automated Reasoning* group of the Department of Computer Science of University of Turin in which I worked and, in particular, dr. *Maria Luisa Sapino.* I would like to thank the reviewers, prof. *Mariangiola Dezani* (University of Turin), prof. *Paola Mello* (University of Ferrara), and prof. *Camilla Schwind* (Laboratoire d'Informatique de Marseille), for their precious advice.

I would like to thank also *Cristina Baroglio*, all my friends at the Department of Computer Science, especially *Ferruccio Damiani* and *Davide Cavagnino*, and all those persons that with their love and their support made this thesis possible, in particular, *my family.*

Turin, Italy M. B.
February 1998

Part One

Inclusion Modal Logics

Chapter I

Introduction

Among true propositions, sometimes it is useful to distinguish between those that are *occasionally* true and those that are *necessarily* true; for instance, a proposition could be true in a particular scenario while another must be true in any possible scenario. Modal logic extends classical logic allowing the occurrence of a new operator, usually denoted by $\Box$, in front of formulae. Differently than the others such as negation or implication, this operator is not intended to be *truth-functional*, i.e. its meaning does not depend only on the truth-values of the subformulae. Indeed, the intended meaning of the formula $\Box\varphi$ is to *qualify* the truth value of φ: if φ is true then, $\Box\varphi$ specifies that φ is not only true but *necessarily true*, i.e. φ is true independently from the scenario (or state, world, etc.) [Hughes and Cresswell, 1996].

Multimodal logics generalize modal logics allowing more than one modal operator to appear in front of formulae. In particular, a modal operator is named by means of a label, for instance $[a]$, which identifies it. Multimodal logics are particularly suitable to reason in a multiagent environment, to represent *knowledge*, *beliefs* and, then, also common interpretation of a formula like $[a]\varphi$ is "φ is *known* by the agent a", "φ is part of the knowledge of a", and "φ is *believed* by the agent a" but also "φ is true after executing the *action* a" [Halpern and Moses, 1992].

The meaning of *necessity* is different depending on the *properties* that one ascribes it. For example, one can say that everything that is necessarily true is also true while another can think that everything that is necessary is necessarily necessary. Moreover, in the multimodal case, modal operators do not represent only necessity but also knowledge, beliefs, actions, etc. It is easy to express the properties which characterize a modal operator by means of a set of axioms. Let us consider, for instance, the modal operator $[a]$. Then, the axiom

$$T(a) : [a]\varphi \supset \varphi$$

(the *knowledge axiom* or *reflexivity*) can express the fact that everything that is necessarily true is also true but also that what is known by the agent a must be true, while the axiom

$$4(a) : [a]\varphi \supset [a][a]\varphi$$

(the *positive introspection axiom* or *transitivity*) can express the fact that everything that is necessary is necessarily necessary, but also that if something is know by a then a knows that he knows it. Furthermore, by using more than one modal operator, we are also able to express what an agent knows (believes) about the knowledge (beliefs) of other agents. For example, the formula $[a][b]\alpha$ can be read as "*the agent a knows (believes) that the agent b knows (believes)* α". Moreover, we can define modal systems characterized by means of *interaction* axioms, such as, for instance,

$$I(a, b) : [a]\varphi \supset [b]\varphi$$

that say that whatever the agent a knows (believes), the agent b knows (believes), the *persistence axiom*

$$P(a, always) : [a][always]\varphi \supset [always][a]\varphi$$

that says that the agent a knows (believes) that φ holds always then a will always know φ, and the *mutual transitivity* axiom

$$4M(always, a) : [always]\varphi \supset [a][always]\varphi$$

to express the fact, for instance, that if something always holds it always also holds after executing the action a.

As pointed out in [Catach, 1988], the main feature of multimodal systems is their ability to express *complex modalities*, obtained by composing modal operators of different types. Thus, such systems allow one to design agent situations where the agents can have different ways of reasoning and different ways of interacting between them and, also, to simultaneously study several modal aspects (e.g., *knowledge and time* or *knowledge and belief* [Catach, 1991]).

Let us consider the following example inspired by [Fariñas del Cerro and Herzig, 1995]. It shows a multimodal system with modalities representing *actions* and *beliefs* of agents and it is based on the fable "the fox and the raven", in which the fox tries to capture the raven's cheese. In order to do so the fox charms the raven.

Example I.0.1 *(The fox and the raven)* Let $[fox]$ be a modal operator axiomatized by only the axiom K and representing what the fox believes and let $[praise]$ and $[sing]$ be two action operators of type K representing the action in which the fox praises the raven and the raven sings, respectively. Moreover, we have a operator $[always]$ of type $KT4$:

(A_1) $T(always) : [always]\varphi \supset \varphi$
(A_2) $4(always) : [always]\varphi \supset [always][always]\varphi$

for which we assume the *mutual transitivity* axioms:

(A_3) $4M(always, praise) : [always]\varphi \supset [praise][always]\varphi,$
(A_4) $4M(always, sing) : [always]\varphi \supset [sing][always]\varphi,$

in order to express the fact that if φ is always true then it is also always true after the actions *prise* and *sing*. We have the following:

(1) $[fox][praise]charmed(raven)$
(2) $[fox][always](charmed(raven) \supset \langle sing\rangle dropped(cheese))$

That is, (1) the fox believes that if the fox praises the raven, then the raven is charmed, and (2) the fox believes that in any moment if the raven is charmed then it is possible that the raven sings and so it drops the cheese. From (1) and (2), the formula:

(3) $[fox][praise]\langle sing\rangle dropped(cheese)$

can be proved; that is, the fox believes that after praising the raven may sing and so it drops the cheese.

In this thesis we work on *normal multimodal logics*, that are general modal systems with an arbitrary set of normal modal operators all characterized by the axiom

$$K(a) : [a](\varphi \supset \psi) \supset ([a]\varphi \supset [a]\psi)$$

focusing on the class of *inclusion modal logics*. This class of logics includes some well-known modal systems such as K_n, T_n, $K4_n$, and $S4_n$. However, differently than other proposals, such as [Halpern and Moses, 1992], these systems can be *non-homogeneous* (i.e., every modal operator is not restricted to the same system) and can contain some *interaction axioms* (i.e., every modal operator is not necessarily independent from the others).

In particular, inclusion modal logics are characterized by sets of logical axioms of the form:

$$[t_1][t_2]\dots[t_n]\varphi \supset [s_1][s_2]\dots[s_m]\varphi \quad (n > 0,\ m \geq 0)$$

that we call *inclusion axioms*. The knowledge axiom, positive introspection axiom, the axiom $I(a, b)$, the mutual transitivity axiom and the persistence axiom are examples of inclusion axiom schema. The syntax, the possible-world semantics, and the axiomatization of inclusion modal logics will be introduced in Chapter II.

Inclusion modal logics have interesting computational properties because they can be considered as rewriting rules. More precisely, inclusion modal logics have been introduced in [Fariñas del Cerro and Penttonen, 1988] with the name of grammar logics to the aim of simulating the behaviour of grammars by means of modal logics. Intuitively, given a *formal grammar*, we associate an axiom of a modal logic to each rule. The idea is quite simple. For each production of the form $t_1 \dots t_n \rightarrow s_1 \dots s_m$ a new inclusion axiom $[t_1]\dots[t_n]\varphi \supset [s_1]\dots[s_m]\varphi$ is introduced. By this construction, verifying if a word is generated by a formal grammar is equivalent to proving a theorem in the logic. As a consequence, the authors of [Fariñas del Cerro and Penttonen, 1988] obtained a simple proof of undecidability for propositional modal logics. However, they neither prove any *(un)decidability* results for restricted classes nor they consider any *proof method* to deal with the whole class of logics (or its subclasses). More recently, in [Gasquet, 1994; Gasquet, 1993], an optimized *functional translation* method for translating formulae of the inclusion modal logics into formulae of the classical first order logic is proposed when, however, the seriality is assumed for each operator.

In this part of the thesis, we answer to the open problems left in [Fariñas del Cerro and Penttonen, 1988]. We first develop an *analytic tableau calculus* for the class of inclusion modal logics and, then, we use it as a tool to prove some *undecidability* results for some subclasses of inclusion modal logics.

Although an axiom system is a calculus, it is not an appropriate choice for automation because, in general, it is hard to find a proof for a given formula, especially in an automatic way. The reason is that axiom systems make use of the modus ponens rule so that to prove a formula φ we have to look for a prove of ψ and $\psi \supset \varphi$ and, generally, ψ may be an arbitrary formula without any relation with φ. Other calculi, such as *resolution*, *sequent calculus*, and *tableau calculus* better work towards this purpose. The fact is that these methods use the "*subformula principle*": everything you need to prove or disprove a given formula is contained in the formula itself.

Among the above mentioned calculi, we have chosen to develop a tableau method in order to supply a proof theory for the class of inclusion modal logics. A tableau calculus is a *refutation method*; given a formula, say φ, the computation process is aimed at finding an *interpretation* which satisfies φ. Consequently, to fail in finding an interpretation which satisfies the negation of φ ($\neg\varphi$) corresponds to prove that φ is true in every interpretation, i.e. φ is *valid*.

There are several reasons that have leaded to prefer developing a tableau calculus instead of a resolution calculus (sequent calculus can be seen as a notational variant of tableau calculus) to study inclusion modal logics. First of all, it does not require any *normal forms*, so the starting formula can use all connectives. Moreover, due to the strong relationship with the semantics issue, tableau calculi are easier and more natural to develop especially for non-classical logics for which, generally, the semantics is known better than the computational properties [Fitting, 1983]. Last but not least, tableau methods enjoy another important feature with respect to resolution: they can supply a *return answer*. Besides the success or the failure, the tableau method returns some more information. In the case of success, it returns an effective interpretation that satisfies the given formula while, in the case of failure, it shows why it is not possible to satisfy that formula by means of an effective contradictory interpretation.

The tableau calculus, presented in Chapter III, is an extension of the one proposed in [Nerode, 1989], which is closely related to the systems of *prefixed tableaux* presented in [Fitting, 1983].

Prefixed tableaux, differently than other tableau methods, make explicit reference to the possible-worlds of the underlying Kripke interpretation. However, as a difference with [Fitting, 1983], worlds are not represented by prefixes (which describe paths in the model from the initial world), instead, they are given an atomic name and the accessibility relationships among them are explicitly represented in a graph. The method is based on the idea of using the characterizing axioms of the logics as "*rewrite rules*" which create new paths among worlds in the counter-model construction.

We think that the tableau calculus is interesting, first, because it is modular with respect to the inclusion modal systems considered, that is it works for the whole class of inclusion modal logics. Then, it deals with *non-homogeneous* multimodal systems with *arbitrary* interaction axioms in an uniform way.

The proposals in [Governatori, 1995; Cunningham and Pitt, 1996; Beckert and Goré, 1997] address the problem of an efficient implementation of the tableau calculi for a wide class of modal logics. They generalize the prefixes by allowing occurrences of variables and they use unification to show that two prefixes are names for the same world. While a straightforward implementation of our calculus is unlikely to be efficient, the generality of the approach makes it suitable to study the properties of different classes of logics. In particular, our tableau calculus is at the basis of the *undecidability* results for inclusion modal logics presented in Chapter IV. Due to the fact that the accessibility relationships among the worlds are represented in a graph and that we use the axioms of the logics as rewrite rules to create new paths among worlds in the counter-model construction, our tableau method allows to draw some correspondences between logic and *formal languages*. These allow to reduce in a easy way some undecidability results of the formal languages to satisfiability problems in the logic.

A *decidability* result for a particular subclass of the inclusion modal logics is also given. This result is obtained by means of the *filtration method*, defining an extension of the Fischer-Ladner closure [Fischer and Ladner, 1979].

Finally, in Chapter V, the tableau method is extended in order to deal with the predicative case, while in Chapter VI, it is shown how our tableau calculus can be easily extended in order to deal with the class of normal multimodal logics generated by the *interaction axiom schemas*

$$G^{a,b,c,d} : \langle a\rangle[b]\varphi \supset [c]\langle d\rangle\varphi$$

proposed in [Catach, 1988], where $\langle a\rangle$ is the modal operator defined as $\neg[a]\neg$. This class includes the class on inclusion modal logics and most of the well-known modal logics studied in [Chellas, 1980; Hughes and Cresswell, 1996] and their multimodal version in [Halpern and Moses, 1992].

Chapter II

Syntax and Semantics

In this chapter we introduce the class of *inclusion modal logics*. We use the world "inclusion" because the logics of this class are characterized by axiom systems whose axioms determine a set of inclusion relations between the accessibility relations of their possible-worlds semantics.

Many results reported in this chapter can be easily deduced from well-known works in literature. Nevertheless, for completeness, we will present them, avoiding to report the most trivial steps.

II.1 Syntax

Let us define a language for a *propositional multimodal logic*. Although we consider a number of different logics in the following, the syntax for all of them is essentially the same. The *alphabet* contains:

- a non-empty countable set VAR of *propositional variables*;
- a non-empty countable set MOD, named the *modal alphabet*. VAR and MOD are disjoint;
- the *classical connectives* " $\wedge$ " (*and*), "$\vee$" (*or*), "$\neg$" (*not*), " $\supset$ " (*implies*);
- a *modal operator constructor* "[.]";
- left and right *parentheses* "(", ")".

The set FOR of formulae of a modal propositional language $\mathcal{L}$ is defined to be the least set that satisfies the following conditions:

- VAR $\subseteq$ FOR;
- if $\varphi, \psi \in$ FOR then $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \supset \psi) \in$ FOR;
- if $\varphi \in$ FOR and $t \in$ MOD then $([t]\varphi) \in$ FOR.

For readability, we omit parentheses if they are unnecessary: we give " $\wedge$ " and "$\vee$" the same precedence; lower that "$\neg$" but higher than " $\supset$ ". Moreover, we use the standard abbreviation $\langle t\rangle\varphi$ for $\neg[t]\neg\varphi$. $[t]$ is called *universal modal operator* or *universal modality*, while $\langle t\rangle$ is called *existential modal operator* or *existential modality*. By *atomic formula* we mean any propositional variables of VAR.

We call $\mathcal{I}_{\mathcal{L}}$ the propositional multimodal logic based on the a language $\mathcal{L}$.

II.2 Possible-worlds semantics

Given a language $\mathcal{L}$, an ordered pair $(W, \{\mathcal{R}_t \mid t \in \text{MOD}\})$, consisting of a non-empty set W of "*possible worlds*" and a set of *binary relations* $\mathcal{R}_t$ (one for each $t \in \text{MOD}$) on W, is called *frame*. Note that frames with an infinite number of possible worlds in W are allowed. We say that w' is *accessible* from w *by means of* $\mathcal{R}_t$ if $(w, w') \in \mathcal{R}_t$, $\mathcal{R}_t$ is the *accessibility relation* of the modality $[t]$. We denote with $\mathcal{F}_{\mathcal{L}}$ the class of all frames based on the language $\mathcal{L}$.

In order to define the meaning of a formula, we have to introduce the notion of *Kripke interpretation*.

Definition II.2.1 (Kripke interpretation) *Given a language $\mathcal{L}$, a Kripke interpretation M is an ordered triple $\langle W, \{\mathcal{R}_t \mid t \in \text{MOD}\}, V\rangle$, where:*

- $(W, \{\mathcal{R}_t \mid t \in \text{MOD}\})$ *is a frame of $\mathcal{F}_{\mathcal{L}}$;*
- *V is a* valuation function, *a mapping from $W \times \text{VAR}$ to the set $\{\mathbf{T}, \mathbf{F}\}$.*

We say that M is based on *the frame $(W, \{\mathcal{R}_t \mid t \in \text{MOD}\})$.*

We use $\mathcal{M}_{\mathcal{L}}$ to denote the class of Kripke interpretations with $\mathcal{L}$ as underlying language.

The meaning of a formula belonging to $\mathcal{L}$ is given by means of the *satisfiability relation* $\models$. In particular, let $M = \langle W, \{\mathcal{R}_t \mid t \in \text{MOD}\}, V\rangle$ be a Kripke interpretation, w a world in W and φ a formula, then, we say that φ is *satisfiable* in the Kripke interpretation M *at* w, denoted by $M, w \models \varphi$, if the following conditions hold:

- $M, w \models \varphi$ and $\varphi \in \text{VAR}$ iff $V(w, \varphi) = \mathbf{T}$;
- $M, w \models \neg\varphi$ iff $M, w \not\models \varphi$;
- $M, w \models \varphi \wedge \psi$ iff $M, w \models \varphi$ and $M, w \models \psi$;
- $M, w \models \varphi \vee \psi$ iff $M, w \models \varphi$ or $M, w \models \psi$;
- $M, w \models \varphi \supset \psi$ iff $M, w \not\models \varphi$ or $M, w \models \psi$;
- $M, w \models [t]\varphi$ iff for all $w' \in W$ such that $(w, w') \in \mathcal{R}_t$, $M, w' \models \varphi$;
- $M, w \models \langle t\rangle\varphi$ iff there exists a $w' \in W$ such that $(w, w') \in \mathcal{R}_t$ and $M, w' \models \varphi$.

Given a Kripke interpretation $M = \langle W, \{\mathcal{R}_t \mid t \in \text{MOD}\}, V\rangle$, we say that a formula φ is *satisfiable in M* if $M, w \models \varphi$ for some world $w \in W$. We say that φ is *valid in M* if $\neg\varphi$ is not satisfiable in M (or, equivalently, if $M, w \models \varphi$, for all worlds in W). Moreover, a formula φ is *satisfiable with respect to a class $\mathcal{M}$* of Kripke interpretations if φ is satisfiable in some Kripke interpretation in $\mathcal{M}$, and it is *valid with respect to $\mathcal{M}$* if it is valid in all Kripke interpretations in $\mathcal{M}$.

II.3 Axiomatization

It is possible to define an axiom system whose axioms and rules of inference characterizes a propositional multimodal logic $\mathcal{I}_{\mathcal{L}}$. In particular, this axiom system, that we call $\mathcal{S}_{\mathcal{L}}$, consists of:

- all axiom schemas for the propositional calculus;
- for each $t \in \text{MOD}$, the axiom schema:

$$K(t) : [t](\varphi \supset \psi) \supset ([t]\varphi \supset [t]\psi)$$

- the *modus ponens* rule of inference: from $\vdash \varphi$[1] and $\vdash \varphi \supset \psi$ infer $\vdash \psi$;
- for each $t \in \text{MOD}$, the *necessitation* rule of inference: from $\vdash \varphi$ infer $\vdash [t]\varphi$.

Each modal system that contains the schema $K(t)$ for each its modal operator is called *normal*. In this thesis we deal with only normal modal logics and its extensions.

The axiomatization $\mathcal{S}_{\mathcal{L}}$ of the propositional modal logic $\mathcal{I}_{\mathcal{L}}$ is *sound* and *complete* with respect to its possible-worlds semantics $\mathcal{M}_{\mathcal{L}}$ [Hughes and Cresswell, 1996; Halpern and Moses, 1992]. Every formula *provable* from $\mathcal{S}_{\mathcal{L}}$ (*$\mathcal{S}_{\mathcal{L}}$-provable*) is valid with respect to $\mathcal{M}_{\mathcal{L}}$ (*soundness*) and every formula that is valid with respect to $\mathcal{M}_{\mathcal{L}}$ is provable from $\mathcal{S}_{\mathcal{L}}$ (*completeness*). We say that a Kripke interpretation M is a *model of $\mathcal{I}_{\mathcal{L}}$* if every $\mathcal{S}_{\mathcal{L}}$-provable formula is valid in M, and F is a *frame for $\mathcal{I}_{\mathcal{L}}$* if every Kripke interpretation based on it is a model of $\mathcal{I}_{\mathcal{L}}$.

Inclusion axiom schemas

An axiom system $\mathcal{S}_{\mathcal{L}}$ can be extended by adding one or more extra axiom schemas. In the following, we are interested in a particular class of such extensions, that is those ones that are obtained by adding only axiom schemas of the following form:

$$[t_1][t_2]\ldots[t_n]\varphi \;\supset\; [s_1][s_2]\ldots[s_m]\varphi \quad (n > 0, m \geq 0)$$

where $t_i, s_j \in \text{MOD}$. We call such an axiom schema *inclusion axiom schema* or *inclusion axiom* for simplicity.

[1]We write $\vdash \varphi$ to mean that φ is a theorem of $\mathcal{S}_{\mathcal{L}}$.

Example II.3.1 Some examples of inclusion axiom schemas are:

- the *knowledge axiom* $T(t) : [t]\varphi \supset \varphi$,
- the *positive introspection axiom* $4(t) : [t]\varphi \supset [t][t]\varphi$,
- the *inclusion axiom* $I(t,t') : [t]\varphi \supset [t']\varphi$,
- the *mutual transitivity axiom* $4M(t,t') : [t]\varphi \supset [t'][t]\varphi$,
- the *persistence axiom* $P(t,t') : [t][t']\varphi \supset [t'][t]\varphi$ [Fariñas del Cerro and Herzig, 1995].

Given a set $\mathcal{A}$ of inclusion axiom schemas, we show that if the accessibility relations in the Kripke interpretations are restricted in a suitable way, the axiom system $\mathcal{S}_{\mathcal{L}}$ extended with $\mathcal{A}$, denoted by $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$, is *sound* and *complete* with respect to possible-worlds semantics. We use $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ to denote the *inclusion propositional modal logic* determined by means of $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$.

Example II.3.2 Some examples of inclusion modal logics are the well-known modal systems K, T, $K4$, $S4$ [Hughes and Cresswell, 1996], their multimodal versions K_n, T_n, $K4_n$, $S4_n$ [Halpern and Moses, 1992], extensions of $S4_n$ with interaction axioms or with agent "any fool" [Genesereth and Nilsson, 1987; Enjalbert and Fariñas del Cerro, 1989].

Remark II.3.1 The class of propositional inclusion modal logics is included in the class of multimodal logics studied in [Catach, 1988]. There, the author generalizes to the multimodal case the *k, l, m, n-incestuality* axiom schema $G^{k,l,m,n} : \Diamond^k \Box^l \varphi \supset \Box^m \Diamond^n$ (see [Chellas, 1980, Section 3.3 and 5.5] and [Hughes and Cresswell, 1984, Chapter 3]). He characterizes the class of modal logics by considering systems axiomatized by any finite number of axiom schemas of the form $G^{a,b,c,d} : \langle a \rangle [b]\varphi \supset [c]\langle d \rangle$, where $\langle a \rangle$, $[b]$, $[c]$, $\langle d \rangle$ can represent sequences of modalities of that type. Thus, when we take into account only axiom schemas of the form $G^{\varepsilon,b,c,\varepsilon}$ we have the class of inclusion modal logics (see Chapter VI for more details).

Some examples

In this section we give an idea of how to use inclusion modal logics to perform *epistemic reasoning* (Example II.3.3, II.3.4, and I.0.1) and to represent simple *reasoning about actions* (Example II.3.5).

In the Examples II.3.3, II.3.4, and I.0.1 we use modal operator to denote *knowledge* and *belief* of agents: a preposition $[t]\varphi$ is read as *"agent t knows φ"* or *"agent t believes φ"*. Inclusion axiom schemas are used to model the meaning of the operator, for example, a modal operator of belief is characterized by only the axiom K, while a modal operator of knowledge by $KT4$ (see [Genesereth and Nilsson, 1987, Chapter 9]). Inclusion axioms are also used to model *interaction* between knowledge or beliefs of different agents. For instance, the axiom $I(t,t') : [t]\varphi \supset [t']\varphi$ can be interpreted as *"everything which is known (believed) by agent t is also known (believed) by agent t'."*

Example II.3.3 *(Epistemic reasoning: The friends puzzle)* Peter is a friend of John, so if Peter knows that John knows something then John knows that Peter knows the same thing. That is, we assume the *persistence* axiom:

$(A_1)\quad P(peter, john) : [peter][john]\varphi \supset [john][peter]\varphi,$

where $[peter]$ and $[john]$ are modal operators of type $S4$ ($KT4$):

$(A_2)\quad T(peter) : [peter]\varphi \supset \varphi;$
$(A_3)\quad 4(peter) : [peter]\varphi \supset [peter][peter]\varphi;$
$(A_4)\quad T(john) : [john]\varphi \supset \varphi;$
$(A_5)\quad 4(john) : [john]\varphi \supset [john][john]\varphi;$

and they are used to denote what is known by Peter and John, respectively. Peter is married, so if Peter's wife knows something, then Peter knows the same thing, that is the *inclusion* axiom:

$(A_6)\quad I(wife(peter), peter) : [wife(peter)]\varphi \supset [peter]\varphi$

holds, where $[wife(peter)]$ is a modality of type $S4$ representing the knowledge of Peter's wife:

$(A_7)\quad T(wife(peter)) : [wife(peter)]\varphi \supset \varphi;$
$(A_8)\quad 4(wife(peter)) : [wife(peter)]\varphi \supset [wife(peter)][wife(peter)]\varphi.$

Thus, we consider a modal language containing three modalities, $[peter]$, $[john]$, and $[wife(peter)]$, and characterized by the set $\mathcal{A} = \{A_i \mid i = 1, \ldots 8\}$ of inclusion axiom schemas.

John and Peter have an appointment, let us consider the following situation:

(1) $[peter]time$
(2) $[peter][john]place$
(3) $[wife(peter)]([peter]time \supset [john]time)$
(4) $[peter][john](place \wedge time \supset appointment)$

That is, (1) Peter knows the time of their appointment; (2) Peter also knows that John knows the place of their appointment. Moreover, (3) Peter's wife knows that if Peter knows the time of their appointment, then John knows that too (since John and Peter are friends); and finally (4) Peter knows that if John knows the place and the time of their appointment, then John knows that he has an appointment. From this situation we will be able to prove:

(5) $[john][peter]appointment \wedge [peter][john]appointment,$

that is, each of the two friends knows that the other one knows that he has an appointment.

In the following example a particular modality is introduced as a certain kind of *common knowledge operator*. Indeed, this modality can be taken as a slightly weaker version of the common knowledge operator in [Halpern and Moses, 1992]. It is slightly weaker because the induction axiom for the common knowledge does not hold [Genesereth and Nilsson, 1987] (see also Remark VI.2.1). The common knowledge operator is achieved using a *fictitious knower*, sometimes called *any fool*. What *any fool* knows is what all other agents know, and all agents know that others know (and so on). In other words, instead of regarding common knowledge as an operator over beliefs of agents, it is regarded as a *new agent* which interacts with the others.

Example II.3.4 *(Epistemic reasoning and common knowledge: The wise men puzzle)* The problem is as follows: "Once upon a time, a king wanted to find the wisest out of his three wisest men. He arranged them in a circle and told them that he would put a white or a black spot on their foreheads and that one of the three spots would certainly be white. The three wise men could see and hear each other but, of course, they could not see their faces reflected anywhere. The king, then, asked to each of them to find out the colour of his own spot. After a while, the wisest correctly answered that his spot was white."

Let us assume a, b, and c to denote the three wise men and by modalities $[a]$, $[b]$, and $[c]$ their beliefs. Moreover, we use $[fool]$ to denote which are known by all the others (the "any fool" agent). Thus, the set of inclusion axioms consists of:

(A_1) $T(fool) : [fool]\varphi \supset \varphi;$
(A_2) $4(fool) : [fool]\varphi \supset [fool][fool]\varphi;$
(A_3) $I(fool, a) : [fool]\varphi \supset [a]\varphi;$
(A_4) $I(fool, b) : [fool]\varphi \supset [b]\varphi;$
(A_5) $I(fool, b) : [fool]\varphi \supset [c]\varphi.$

The modal operators $[a]$, $[b]$, $[c]$, and $[fool]$ give a way to distinguish among information of the single agents and information common to all of them. The formulation is the following, however, in order to avoid introducing many variant of the same formulae for the different wise men, as a shorthand, we use the metavariables X, Y and Z, where $X, Y, Z \in \{a, b, c\}$ and $X \neq Y$, $Y \neq Z$, and $X \neq Z$:

(1) $[fool](\neg ws(X) \land \neg ws(Y) \supset ws(Z))$
(2) $[fool](\neg ws(X) \supset [Y]\neg ws(X))$

$ws(X)$ means X has a white spot on his forehead. All the formulae preceded by the modal operator $[fool]$, correspond to information which is *common* to all wise men. The formula (1) says that at least one of the wise men has a white spot, whereas formula (2) means that whenever one of them has not a white spot, the others know this since the three wise men can see each other. From (1) and (2) we cannot prove $[X]ws(X)$ for any wise man.

Now, the king asks if someone knows if the color of his spot is white, but nobody says anything, therefore X knows that Y does not know the color of his own spot:[2]:

(3) $[X]\neg[Y]ws(Y)$

From (1)-(3) we cannot yet prove $[X]ws(X)$ for any wise man. The king asks again if someone knows if the color of his spot is white, but nobody still say anything, therefore X knows that Y knows that Z does not know the color of his own spot:

(4) $[X][Y]\neg[Z]ws(Z)$

[2]This fact allows to refuse to believe there is only one white spot, otherwise the wise man who has that white spot could have answered (the king said there is at least one white spot).

Now, from (1)-(4) we can prove $[X]ws(X)$ for any wise man: each of them has enough information for answering that he knows that the color of his spot is white[3], but only the wisest will announce that his spot is white.

In the following example, inspired from [Fariñas del Cerro and Herzig, 1995], it is shown how modalities can be used to represent actions. Here the previous common knowledge operator $[fool]$ is used to represent something that holds in any moment, after any sequence of actions. For this reason, now, we call it $[always]$.

Example II.3.5 (*Reasoning about actions: A simple version of the shooting problem*) Assume that our language contains the modalities $[load]$ and $[shoot]$ which denote the actions of "loading a gun" and "shooting against a turkey", respectively, and $[always]$ denoting an arbitrary sequence of actions, where $[always]\varphi$ means that φ always holds (i.e., after any sequence of actions). The set $\mathcal{A}$ will contain the following axioms:

(A_1) $T(always) : [always]\varphi \supset \varphi;$
(A_2) $4(always) : [always]\varphi \supset [always][always]\varphi;$
(A_3) $I(always, load) : [always]\varphi \supset [load]\varphi;$
(A_4) $I(always, shoot) : [always]\varphi \supset [shoot]\varphi;$

Notice that $[always]$ is reflexive (axiom A_1), transitive (axiom A_2), and if φ is always true it is true after the action *load* or *shoot* (axioms A_3 and A_4, respectively), whereas the modalities representing actions do not have any property beside K. Let us assume the situation:

(1) $[always][load]loaded$
(2) $[always](loaded \supset [shoot]\neg alive)$

That is, (1) after any sequence of actions ended by *load* the gun is *loaded*, and (2) after any sequence of actions (possible empty) if the gun is *loaded* then after a *shoot* the turkey is not *alive*. Form (1) and (2) we can prove:

(3) $[load][shoot]\neg alive$

that is, after the actions of *load* and *shoot* the turkey is not *alive*.

Inclusion frames and Kripke $\mathcal{A}$-interpretation

Definition II.3.1 (Inclusion frame) *Let $F = (W, \{\mathcal{R}_t \mid t \in \text{MOD}\})$ be a frame of $\mathcal{F}_\mathcal{L}$ and let $\mathcal{A}$ be a set of inclusion axiom schemas, F is an $\mathcal{A}$-*inclusion frame *if and only if for each axiom schema*

$$[t_1][t_2]\ldots[t_n]\varphi \supset [s_1][s_2]\ldots[s_m]\varphi$$

[3]Actually, if they did not answer twice, this is the only possible configuration. If there were a wise man who has a "not-white" spot, say a, he could not have answered but b (or c) could have. They know that it is not possible to have two "not-white" spots and they can see one, then, they can deduce they have both a white spot. On the other hand, this is also the only fair configuration if the king would like to know the wisest.

in $\mathcal{A}$, the following inclusion property *on the accessibility relation holds:*

$$\mathcal{R}_{t_1} \circ \mathcal{R}_{t_2} \circ \ldots \circ \mathcal{R}_{t_n} \supseteq \mathcal{R}_{s_1} \circ \mathcal{R}_{s_2} \circ \ldots \circ \mathcal{R}_{s_m} \tag{II.1}$$

where "$\circ$" means the relation composition $\mathcal{R}_t \circ \mathcal{R}_{t'} = \{(w, w'') \in W \times W \mid \exists w' \in W$ such that $(w, w') \in \mathcal{R}_t$ and $(w', w'') \in R_{t'}\}$[4]*. We call $IP^{\mathcal{A}}_{\mathcal{L}}$ the set of* inclusion properties *of the form (II.1) determined by $\mathcal{A}$.*

We denote with $\mathcal{F}^{\mathcal{A}}_{\mathcal{L}}$ the subset of $\mathcal{F}_{\mathcal{L}}$ that consists of all $\mathcal{A}$-inclusion frames. A Kripke $\mathcal{A}$-interpretation is a Kripke interpretation based on an $\mathcal{A}$-inclusion frame. The set of all Kripke $\mathcal{A}$-interpretations is denoted by $\mathcal{M}^{\mathcal{A}}_{\mathcal{L}}$ and it is a subset of $\mathcal{M}_{\mathcal{L}}$. Moreover, we also say that a formula φ of $\mathcal{L}$ is *$\mathcal{A}$-satisfiable in M* (*$\mathcal{A}$-valid in M*) if $M \in \mathcal{M}^{\mathcal{A}}_{\mathcal{L}}$ and it is satisfiable in M (valid in M). A formula is *$\mathcal{A}$-satisfiable* (*$\mathcal{A}$-valid*) if it is satisfiable (valid) with respect to the class $\mathcal{M}^{\mathcal{A}}_{\mathcal{L}}$ of Kripke $\mathcal{A}$-interpretations and we use the notation $\models_{\mathcal{A}}$ for it.

For the class $\mathcal{M}^{\mathcal{A}}_{\mathcal{L}}$ of modal Kripke $\mathcal{A}$-interpretations and the satisfiability relation $\models_{\mathcal{A}}$ the following important proposition holds.

Proposition II.3.1 *Given a language $\mathcal{L}$, for all formulae $\varphi, \psi \in$ FOR, all Kripke $\mathcal{A}$-interpretations $M = \langle W, \{\mathcal{R}_t \mid t \in \text{MOD}\}, V\rangle$ of $\mathcal{M}^{\mathcal{A}}_{\mathcal{L}}$, and all worlds $w \in W$ the following properties hold:*

1. *if φ is an instance of a propositional tautology, then $M, w \models_{\mathcal{A}} \varphi$;*
2. *if $M, w \models_{\mathcal{A}} \varphi$ and $M, w \models_{\mathcal{A}} \varphi \supset \psi$, then $M, w \models_{\mathcal{A}} \psi$;*
3. *$M, w \models_{\mathcal{A}} [t](\varphi \wedge \psi) \supset ([t]\varphi \wedge [t]\psi)$;*
4. *for all inclusion axiom schemas $[t_1][t_2]\ldots[t_n]\varphi \supset [s_1][s_2]\ldots[s_m]\varphi$ in $\mathcal{A}$, $M, w \models_{\mathcal{A}} [t_1][t_2]\ldots[t_n]\varphi \supset [s_1][s_2]\ldots[s_m]\varphi$.*

Proof. We report only the proof for the property (4), for the others you can see [Halpern and Moses, 1992, page 325]. Let us assume that $M, w \models_{\mathcal{A}} [t_1][t_2]\ldots[t_n]\varphi$ but $M, w \not\models_{\mathcal{A}} [s_1][s_2]\ldots[s_m]\varphi$. Then, $M, w \models_{\mathcal{A}} \neg[s_1][s_2]\ldots[s_m]\varphi$ and, therefore, there exist $w_1, w_2, \ldots, w_{m-1}, w'$ in W such that $(w, w_1) \in \mathcal{R}_{s_1}$, $(w_1, w_2) \in \mathcal{R}_{s_2}$, $\ldots$, $(w_{m-1}, w') \in \mathcal{R}_{s_m}$ (i.e., $(w, w') \in \mathcal{R}_{s_1} \circ \mathcal{R}_{s_2} \circ \ldots \circ \mathcal{R}_{s_m}$) and $M, w' \models_{\mathcal{A}} \neg\varphi$. Now, since $M \in \mathcal{M}^{\mathcal{A}}_{\mathcal{L}}$ by hypothesis and, therefore, the (II.1) holds, $(w, w') \in \mathcal{R}_{t_1} \circ \mathcal{R}_{t_2} \circ \ldots \circ \mathcal{R}_{t_m}$, thus, there exist $w'_1, w'_2, \ldots, w'_{n-1}$ in W such that $(w, w'_1) \in \mathcal{R}_{t_1}$, $(w'_1, w'_2) \in \mathcal{R}_{t_2}$, $\ldots$, $(w'_{n-1}, w') \in \mathcal{R}_{t_n}$ and $M, w' \models_{\mathcal{A}} \neg\varphi$, but this is contradictory with the initial hypothesis $M, w \models_{\mathcal{A}} [t_1][t_2]\ldots[t_n]\varphi$. □

Remark II.3.2 It is worth noting that inclusion frames do not allow *backward moves*: neither symmetry nor euclideanness determine inclusion frames.

[4]If $m = 0$ then we assume $\mathcal{R}_{s_1} \circ \mathcal{R}_{s_2} \circ \ldots \circ \mathcal{R}_{s_m} = I$, where I is the identity relation on W.

Soundness and completeness

The following theorem states that the axiom system $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$ characterizes the class $\mathcal{M}_{\mathcal{L}}^{\mathcal{A}}$ of Kripke $\mathcal{A}$-interpretations. The proof uses a well-known technique that shows the close correspondence between an axiom system and a particular interpretation, named *canonical model* [Hughes and Cresswell, 1996; Halpern and Moses, 1992]. It is very close to the one given in [Fariñas del Cerro and Penttonen, 1988] for a subclass of the inclusion modal logics, called *Thue logics*.

Theorem II.3.1 *Let $\mathcal{L}$ be a modal language and let $\mathcal{A}$ be a set of inclusion axiom schemas, $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$ is a sound and complete axiomatization with respect to $\mathcal{M}_{\mathcal{L}}^{\mathcal{A}}$.*

Before proving the above theorem, we need to give some definitions and lemmas. A formula φ is *$\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent* if $\neg\varphi$ is not $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-provable. A finite set of formulae is $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent if the conjunction of all them is $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent, and an infinite set of formulae is $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent if all of its finite subsets are $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent. A set S of formulae is *maximal $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent*, if it is $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent and for any formula φ, either $\varphi \in S$ or $\neg\varphi \in S$.

Lemma II.3.1 *Any $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent set of formulae can be extended to a maximal $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent set. Moreover, let S be a maximal $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent set of formulae, then it satisfies the following properties:*[5]

1. *for no formula φ we have $\varphi \in S$ and $\neg\varphi \in S$;*
2. *$\varphi \supset \psi \in S$ if and only if $\neg\varphi \in S$ or $\psi \in S$;*
3. *if $\varphi \in S$ and $\varphi \supset \psi \in S$, then $\psi \in S$;*
4. *if φ is $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-provable, then $\varphi \in S$.*

Proof. See, for a similar proof, [Hughes and Cresswell, 1996, Chapter 6] and [Halpern and Moses, 1992, page 327]. $\square$

Definition II.3.2 (Canonical model) *The canonical model is the ordered triple*

$$\mathcal{M}_c^{\mathcal{A}} = \langle W, \{\mathcal{R}_t \mid t \in \mathrm{MOD}\}, V\rangle$$

where:

- $W = \{w \mid w$ *is a maximal consistent set*$\}$;
- *for each* $t \in \mathrm{MOD}$, $\mathcal{R}_t = \{(w, w') \in W \times W \mid w^t \subseteq w'\}$, *where*

$$w^t = \{\varphi \mid [t]\varphi \in w\}$$

[5]We report the properties only for logical connective "$\neg$" and "$\supset$", the properties for the others can be easily derived.

- *for each* $p \in \mathrm{VAR}$ *and each* $w \in W$*, we set*

$$V(w,p) = \begin{cases} \mathbf{T} & \textit{if } p \in w \\ \mathbf{F} & \textit{otherwise} \end{cases}$$

It is quite easy to see, by the definition of accessibility relations given above, that for any $t, s \in \mathrm{MOD}$ $(w, w') \in \mathcal{R}_t \circ \mathcal{R}_s$ if and only if $(w^t)^s \subseteq w'$, where $(w^t)^s = \{\varphi \mid [t][s]\varphi \in w\}$.

Proposition II.3.2 *The canonical model* $\mathcal{M}_c^{\mathcal{A}}$ *given by Definition II.3.2 is a Kripke* $\mathcal{A}$*-interpretation.*

Proof. We have to prove that each inclusion property in $IP_{\mathcal{L}}^{\mathcal{A}}$ is satisfied by $\mathcal{M}_c^{\mathcal{A}}$. Let us suppose that $\mathcal{R}_{t_1} \circ \ldots \circ \mathcal{R}_{t_n} \supseteq \mathcal{R}_{s_1} \circ \ldots \circ \mathcal{R}_{s_m} \in IP_{\mathcal{L}}^{\mathcal{A}}$, and $(w, w') \in \mathcal{R}_{s_1} \circ \ldots \circ \mathcal{R}_{s_m}$, we have to show $(w, w') \in \mathcal{R}_{t_1} \circ \ldots \circ \mathcal{R}_{t_n}$, that is $(\cdots(w^{t_1})\cdots)^{t_n} \subseteq w'$. Now, let us assume $[t_1]\ldots[t_n]\varphi \in w$ and let us show that $\varphi \in w'$. Since $[t_1]\ldots[t_n]\varphi \supset [s_1]\ldots[s_m]\varphi \in \mathcal{A}$, by Lemma II.3.1(4), $M, w \models_{\mathcal{A}} [t_1]\ldots[t_n]\varphi \supset [s_1]\ldots[s_m]\varphi$. Then, by Lemma II.3.1(2), $[s_1]\ldots[s_m]\varphi \in w$. Therefore, since by hypothesis $(\cdots(w^{s_1})\cdots)^{s_m} \subseteq w'$, we have $\varphi \in w'$. □

Proposition II.3.3 *Let* $\mathcal{M}_c^{\mathcal{A}}$ *be the canonical model given by Definition II.3.2 then, for any formula* φ *and any world* w*,* $\mathcal{M}_c^{\mathcal{A}}, w \models_{\mathcal{A}} \varphi$ *if and only if* $\varphi \in w$*.*

Proof. The proof is by induction of the structure of the formula φ and it is similar to the ones given for the modal systems presented in [Fariñas del Cerro and Penttonen, 1988, page 132], [Halpern and Moses, 1992, page 327], and [Hughes and Cresswell, 1996, Chapter 6]). □

Now, we are in the position to give the proof of the Theorem II.3.1.

Proof. (*of Theorem II.3.1*) *Soundness.* By Preposition II.3.1. *Completeness.* Assume that φ is $\mathcal{A}$-valid and φ is not $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-provable. Then, $\neg\neg\varphi$ is not $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-provable too and, hence, $\neg\varphi$ is $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$-consistent (see page 17). Now, by Lemma II.3.1, $\neg\varphi$ is contained in some maximal consistent set, say w. Thus, by Proposition II.3.3, $\mathcal{M}_c^{\mathcal{A}}, w \models_{\mathcal{A}} \neg\varphi$. But this is a contradiction because we assumed by hypothesis that φ is $\mathcal{A}$-valid. □

Remark II.3.3 It is worth noting that it is not the case that every model for $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$ satisfies $IP_{\mathcal{L}}^{\mathcal{A}}$, even though every Kripke $\mathcal{A}$-interpretation is a model of $\mathcal{S}_{\mathcal{L}}^{\mathcal{A}}$ (Theorem II.3.1).

Example II.3.6 Let us suppose a modal language with $\mathrm{MOD} = \{t, s\}$, $\mathrm{VAR} = \{p\}$ and let $\mathcal{A}$ be the set of inclusion axioms $\{[t]\varphi \supset [s]\varphi\}$. Now, let M be the Kripke interpretation $\langle W, \{\mathcal{R}_t, \mathcal{R}_s\}, V\rangle$, where $W = \{w_1, w_2, w_3\}$, $\mathcal{R}_t = \{(w_1, w_2)\}$, $\mathcal{R}_s = \{(w_1, w_3)\}$, and $V(w_2, p) = V(w_3, p) = \mathbf{T}$. Clearly, since M does not satisfies $IP_{\mathcal{L}}^{\mathcal{A}} = \{\mathcal{R}_t \supseteq \mathcal{R}_s\}$, M is not a Kripke $\mathcal{A}$-interpretation, though it is possible to show that M is a model of $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$.[6]

[6]Before we show that each formula $\varphi \in \mathrm{FOR}$, we have $M, w_2 \models_{\mathcal{A}} \varphi$ iff $M, w_3 \models_{\mathcal{A}} \varphi$ by induction on the structure of φ. Then, it easy to see that for all formula φ and all world $w \in W$, $M, w \models_{\mathcal{A}} [t]\varphi \supset [s]\varphi$ ([Hughes and Cresswell, 1996, Chapter 10]).

Nevertheless, if we look at the level of frame rather than at the level of Kripke interpretation, we can state that $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ is characterized by the class of all frame that satisfy $IP_{\mathcal{L}}^{\mathcal{A}}$.

Theorem II.3.2 *F is a frame for $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ if and only if $F \in \mathcal{F}_{\mathcal{L}}^{\mathcal{A}}$.*

Proof. (*Only if*) By Theorem II.3.1. (*If*) Let $F = (\langle W, \{\mathcal{R}_t \mid t \in \text{MOD}\})$ be a frame of $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ and $F \notin \mathcal{F}_{\mathcal{L}}^{\mathcal{A}}$. Then, for some pair of worlds in W, say w and w', $(w, w') \in \mathcal{R}_{s_1} \circ \ldots \circ \mathcal{R}_{s_m}$ but $(w, w') \notin \mathcal{R}_{t_1} \circ \ldots \circ \mathcal{R}_{t_n}$, such that $[t_1] \ldots [t_n]\varphi \supset [s_1] \ldots [s_m]\varphi \in \mathcal{A}$. Let M be a Kripke $\mathcal{A}$-interpretation based on F in which the valuation function V is defined on $p \in \text{VAR}$ so that $V(w', p) = \mathbf{T}$ and, for all $w'' \in W$ such that $w'' \neq w'$, $V(w'', p) = \mathbf{F}$. Now, since $(w, w') \notin \mathcal{R}_{t_1} \circ \ldots \circ \mathcal{R}_{t_n}$, it is easy to see that $M, w \models [t_1] \ldots [t_n]p$. Moreover, $M, w \models \neg[s_1] \ldots [s_m]p$, hence, $M, w \not\models [t_1] \ldots [t_n]\varphi \supset [s_1] \ldots [s_m]\varphi$. This is a contradiction by Proposition II.3.1. $\Box$

Chapter III

Proof Theory

In this chapter we develop an *analytic tableau calculus* for the class of *propositional inclusion modal logics*. This calculus will be *modular* with respect to the set of inclusion axioms $\mathcal{A}$. The method is based on the idea of using the characterizing axioms of the logic as "*rewrite rules*" which create new paths among worlds in the counter-model construction.

The calculus is an extension of the one proposed in [Nerode, 1989], which is closely related to the systems of prefixed tableaux presented in [Fitting, 1983]. As a difference with [Fitting, 1983], worlds are not represented by prefixes (which describe paths in the model from the initial world), but they are given an atomic name and the accessibility relationships among them are explicitly represented in a graph.

III.1 Preliminary notions

Before introducing our tableau calculus, we need to define some notions. First of all, we define a *signed formula* Z of a language $\mathcal{L}$ as a formula prefixed by one of the two symbols $\mathbf{T}$ and $\mathbf{F}$ (*signs*). For instance, if φ is a formula of $\mathcal{L}$ then, $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ are signed formulae of $\mathcal{L}$.

Definition III.1.1 *Let $\mathcal{L}$ be a propositional modal language and let $\mathcal{W}_C$ be a countable non-empty set of* constant world symbols *(or* prefixes*), a* prefixed signed formula*, $w : Z$, is a prefix $w \in \mathcal{W}_C$ followed by a signed formula Z.*

We assume $\mathcal{W}_C$ contains always at least the prefix i, that is interpreted as the *initial world.*

Definition III.1.2 *Let $\mathcal{L}$ be a propositional modal language, an* accessibility relation formula *$w\ \rho_t\ w'$, where $t \in$ MOD, is a binary relation between constant world symbols of $\mathcal{W}_C$.*

We say that an accessibility relation formula $w\ \rho_t\ w'$ is *true* in a tableau branch if it belongs to that branch. A *tableau* is a *labeled tree* where each node consists of a *prefixed* *signed formula* or of an *accessibility relation formula.* Intuitively, each tableau branch corresponds to the construction of a Kripke interpretation that satisfies the formulae that belong to it. Intuitively, prefixes are used to name worlds; a formula $w : \mathbf{T}\varphi$ ($w : \mathbf{F}\varphi$) on a branch of a tableau means that the formula φ is *true* (*false*) at the world w, in the Kripke interpretation represented by that branch. Moreover, an accessible relation formula $w\ \rho_t\ w'$ true in a tableau branch means that in the Kripke interpretation represented by that branch w' is accessible form w by means of the accessibility relation of $[t]$.

Remark III.1.1 Using prefixed formulae is very common in modal theorem proving (see [Goré, 1995] for an historical introduction on the topic). We would like to mention the well-known *prefixed tableau systems* in [Fitting, 1983] and the TABLEAUX system in [Catach, 1991]. In [Fitting, 1983], differently than our approach and the ones in [Nerode, 1989; Catach, 1991], a prefix is a sequence of integers which represents a world as a *path* from the initial world to it. As a result, instead of representing *explicitly* worlds and accessibility relations of a Kripke interpretation as a *graph*, by means of the accessibility relation formulae, [Fitting, 1983] represents them as a set of paths, which can be considered as a *spanning tree* of the graph. Similar ideas are also used by other authors, such as the proposals in [Massacci, 1994; Governatori, 1995; Cunningham and Pitt, 1996; De Giacomo and Massacci, 1996].

Conjunctive formulae

α	α_1	α_2
$\mathbf{T}(\varphi \wedge \psi)$	$\mathbf{T}\varphi$	$\mathbf{T}\psi$
$\mathbf{F}(\varphi \vee \psi)$	$\mathbf{F}\varphi$	$\mathbf{F}\psi$
$\mathbf{F}(\varphi \supset \psi)$	$\mathbf{T}\varphi$	$\mathbf{F}\psi$
$\mathbf{F}(\neg\varphi)$	$\mathbf{T}\varphi$	$\mathbf{T}\varphi$

Disjunctive formulae

β	β_1	β_2
$\mathbf{F}(\varphi \wedge \psi)$	$\mathbf{F}\varphi$	$\mathbf{F}\psi$
$\mathbf{T}(\varphi \vee \psi)$	$\mathbf{T}\varphi$	$\mathbf{T}\psi$
$\mathbf{T}(\varphi \supset \psi)$	$\mathbf{F}\varphi$	$\mathbf{T}\psi$
$\mathbf{T}(\neg\varphi)$	$\mathbf{F}\varphi$	$\mathbf{F}\varphi$

Necessary formulae

ν^t	ν^t_0
$\mathbf{T}([t]\varphi)$	$\mathbf{T}\varphi$
$\mathbf{F}(\langle t\rangle\varphi)$	$\mathbf{F}\varphi$

Possible formulae

π^t	π^t_0
$\mathbf{F}([t]\varphi)$	$\mathbf{F}\varphi$
$\mathbf{T}(\langle t\rangle\varphi)$	$\mathbf{T}\varphi$

Figure III.1: Uniform notation for propositional signed modal formulae.

In order to simplify the presentation of the calculus and the proofs we use the well-known *uniform notation* for signed formulae. The uniform notation has been introduced by Smullyan in [Smullyan, 1968] and developed and extensively used for the modal logic by Fitting in [Fitting, 1973; Fitting, 1983]. It classifies non-atomic signed formulae according to their sign and main connective. Figure III.1 reports the complete classification for propositional modal formulae of the Chapter II. In the following, we will often use α, β, ν^t, and π^t as formulae of the corresponding type.

III.2 A tableau calculus

A tableau is an attempt to build an interpretation in which a given formula is satisfiable. Starting from a formula φ, the interpretation is progressively constructed applying a set of *extension rules*, which reflect the semantics of the considered logic. At any stage, a branch of a tableau is a partial description of an interpretation. Usually, the tableau methods are used as a *refutation method.* Proving that a formula φ is a theorem of a certain logic means to show that the attempt to satisfy $\neg\varphi$ leads to *contradictory interpretations.*

In our case, the tableau method tries to build Kripke interpretations, one for each branch: the worlds are formed by the prefixes that appear on the branch, the accessibility relations for the modalities are given by means of the accessibility relation formulae, and the valuation function is given by means of the prefixed signed atomic formulae.

Now, we can present the set of extension rules. But, before doing this, we need to introduce some terminology. In particular, we say that a prefix w is *used* on a tableau branch if it occurs on the branch in some accessibility relation formula, otherwise we say that prefix w is *new.*

Definition III.2.1 (Extension rules) *Let $\mathcal{L}$ be a modal language and let $\mathcal{A}$ be a set of inclusion axioms, the extension rules (tableau rules) for $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}}$ are given in Figure III.2.*

$$\frac{w : \alpha}{\begin{array}{c} w : \alpha_1 \\ w : \alpha_2 \end{array}}\ \alpha\text{-rule} \qquad\qquad \frac{w : \beta}{w : \beta_1 \mid w : \beta_2}\ \beta\text{-rule}$$

$$\frac{w : \nu^t \quad w\ \rho_t\ w'}{w' : \nu^t_0}\ \nu\text{-rule} \qquad\qquad \begin{array}{c} \dfrac{w : \pi^t}{\begin{array}{c} w' : \pi^t_0 \\ w\ \rho_t\ w' \end{array}}\ \pi\text{-rule} \\ \text{where } w' \text{ is } \textit{new} \text{ on the branch} \end{array}$$

$$\begin{array}{c} \dfrac{w\ \rho_{s_1}\ w_1 \quad \cdots \quad w_{m-1}\ \rho_{s_m}\ w'}{\begin{array}{c} w\ \rho_{t_1}\ w'_1 \\ \vdots \\ w'_{n-1}\ \rho_{t_n}\ w' \end{array}}\ \rho\text{-rule} \\ \text{where } w'_1, \ldots, w'_{n-1} \text{ are } \textit{new} \text{ on the branch} \\ \text{and } [t_1] \ldots [t_n]\varphi \supset [s_1] \ldots [s_m]\varphi \in \mathcal{A} \end{array}$$

Figure III.2: Tableau rules for propositional inclusion modal logics.

The interpretation of the different kinds of extension rules is rather easy taking into account the possible-worlds semantics (see Section II.2). The rules for the formula of type α and β are the usual ones of classical calculus (a part from the prefixes).

A formula of type ν^t is true at world w if ν^t_0 is true in all world w' accessible from w by means of t. Therefore, if $w : \nu^t$ occurs on an open branch, we can add $w' : \nu^t_0$ at the end of that branch for any w' which is accessible from w by means of the accessible relation associated with the modal operator $[t]$ (i.e., $w \ \rho_t \ w'$ is true in that branch).

A formula of type π^t is true at the world w by means of t if there exists a world w' accessible from w in which π_0^t is true. Therefore, if $w : \pi^t$ occurs on an open branch, we can add $w' : \pi_0^t$ to the end of that branch, provided w' is new and $w \ \rho_t \ w'$ is true in it.

The intuition behind the ρ-rule is quite simple. Let us suppose, for instance, that $[t_1] \dots [t_n]\varphi \ \supset [s_1] \dots [s_m]\varphi \in \mathcal{A}$ is an axiom of our modal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$. If $w = w_0 \ \rho_{s_1} \ w_1, \dots, w_{m-1} \ \rho_{s_m} \ w_m = w'$ are true in a tableau branch then, w_i is accessible from w_{i-1} by means of s_i in the Kripke interpretation represented by that branch. Since $[t_1] \dots [t_n]\varphi \supset [s_1] \dots [s_m]\varphi \in \mathcal{A}$ then, the corresponding inclusion property (II.1) must hold. Thus, there must exist a set of worlds $w = w'_0, w'_1, \dots, w'_{n-1}, w'_n = w'$ such that w'_i is accessible from w'_{i-1} by means of t_i. Thus, we can add the formulae $w \ \rho_{t_1} \ w'_1, \dots, w'_{n-1} \ \rho_{t_n} \ w'$ to that branch provided that $w'_1, \dots, w'_{n-1}$ are new. Note that, in the case of $m = 0$ we can add the formulae $w \ \rho_{t_1} \ w'_1, \dots, w'_{n-1} \ \rho_{t_n} \ w$.

Remark III.2.1 It is worth noting that the ρ-rule works for the whole class of inclusion modal logics as well as the proofs in the next section. This is the advantages of our approach. On the other hand, the proposed tableau calculus can also be thought as being *modular* with respect to different modal logics than inclusion modal logics. Indeed, it can be extended in order to deal a wider class of modal logics as we show in Chapter VI.

We say that a tableau branch is *closed* if it contains $w : \mathbf{T}\varphi$ and $w : \mathbf{F}\varphi$ for some formula φ. A tableau is *closed* if *every* branch in it is closed. Now, we are in the position to define the meaning of *proof*.

Definition III.2.2 *Let $\mathcal{L}$ be a modal language and let $\mathcal{A}$ a set of inclusion axioms. Then, given a formula φ, we say that a closed tableau for $i : \mathbf{F}\varphi$, using the tableau rules of Figure III.2, is a* proof *of φ (we also say that φ is $\mathcal{T}_{\mathcal{L}}^{\mathcal{A}}$-*provable*).*

Example III.2.1 *(The fox and the raven)* We give here the proof of formula (3) from (1) and (2) in Example I.0.1. We use the symbol "×" to say that a tableau branch is closed.

1. $i : \mathbf{T}[fox][praise]charmed(raven)$
2. $i : \mathbf{T}[fox][always](charmed(raven) \supset \langle sing \rangle dropped(cheese))$
3. $i : \mathbf{F}[fox][praise]\langle sing \rangle dropped(cheese)$
4. $w_1 : \mathbf{F}[praise]\langle sing \rangle dropped(cheese)$
5. $i \ \rho_{fox} \ w_1$
6. $w_2 : \mathbf{F}\langle sing \rangle dropped(cheese)$
7. $w_1 \ \rho_{praise} \ w_2$
8. $w_1 : \mathbf{T}[praise]charmed(raven)$
9. $w_2 : \mathbf{T}charmed(raven)$
10. $w_1 : \mathbf{T}[always](charmed(raven) \supset \langle sing \rangle dropped(cheese))$
11. $w_1 \ \rho_{always} \ w_1$
12. $w_1 \ \rho_{always} \ w_2$
13. $w_2 : \mathbf{T}(charmed(raven) \supset \langle sing \rangle dropped(cheese))$

14a. $w_2 : \mathbf{F} charmed(raven)$
$\times$
14b. $w_2 : \mathbf{T}\langle sing\rangle dropped(cheese)$
15b. $w_3 : \mathbf{T} dropped(cheese)$
16b. $w_2 \; \rho_{sing} \; w_3$
17b. $w_3 : \mathbf{F} dropped(cheese)$
$\times$

We denote with "a" and "b" the two branches which are created by the application of β-rule to step 13. Explanation: *1. and 2.*: formula (1) and (2) from Example I.0.1; *3.*: goal, formula (3) from Example I.0.1; *4. and 5.*: from 3., by application of π-rule; *6. and 7.*: from 4., by π-rule; *8.*: from 1. and 5., by ν-rule; *9.*: from 8. and 7., by ν-rule; *10.*: from 2. and 5., by ν-rule; *11.*: by (A_1) and ρ-rule; *12.*: from 7. and 11., by axiom (A_3) and ρ-rule; *13.*: from 10. and 12., by ν-rule; *14a. and 14b.*: from 13., by β-rule, branch "a" closes; *15b. and 16b.*: from 14., by π-rule; *17b.*: from 6 and 16b., by ν-rule, branch "b" closes.

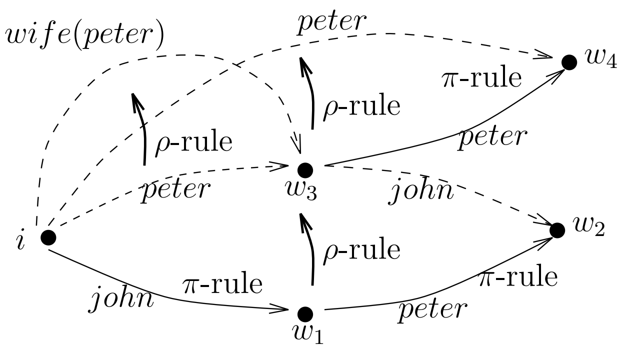

Figure III.3: ρ-rule as rewriting rule: counter-model construction of Example III.2.2.

Example III.2.2 *(The friends puzzle)* We prove the first conjunct of the formula (5) in Example II.3.3 (the proof for the second conjunct is similar) from the set of formulae (1)-(4).

1. $i : \mathbf{T}[peter]time$
2. $i : \mathbf{T}[wife(peter)]([peter]time \supset [john]time)$
3. $i : \mathbf{T}[peter][john]place$
4. $i : \mathbf{T}[peter][john](place \wedge time \supset apointment)$
5. $i : \mathbf{F}[john][peter]appointment$
6. $w_1 : \mathbf{F}[peter]appointment$
7. $i \; \rho_{john} \; w_1$
8. $w_2 : \mathbf{F}appointment$
9. $w_1 \; \rho_{peter} \; w_2$
10. $i \; \rho_{peter} \; w_3$
11. $w_3 \; \rho_{john} \; w_2$

12.	$w_3 : \mathbf{T}[john](place \wedge time \supset appointment)$
13.	$w_2 : \mathbf{T}(place \wedge time \supset appointment)$
14a.	$\quad w_2 : \mathbf{T}appointment$
	$\quad \times$
14b.	$\quad w_2 : \mathbf{F}(place \wedge time)$
15ba.	$\qquad w_2 : \mathbf{F}place$
16ba.	$\qquad w_3 : \mathbf{T}[john]place$
17ba.	$\qquad w_2 : \mathbf{T}place$
	$\qquad \times$
15bb.	$\qquad w_2 : \mathbf{F}time$
16bb.	$\qquad i\ \rho_{wife(peter)}\ w_3$
17bb.	$\qquad w_3 : \mathbf{T}([peter]time \supset [john]time)$
18bba.	$\qquad\quad w_3 : \mathbf{T}[john]time$
19bba.	$\qquad\quad w_2 : \mathbf{T}time$
	$\qquad\quad \times$
18bbb.	$\qquad\quad w_3 : \mathbf{F}[peter]time$
19bbb.	$\qquad\quad w_4 : \mathbf{F}time$
20bbb.	$\qquad\quad w_3\ \rho_{peter}\ w_4$
21bbb.	$\qquad\quad i\ \rho_{peter}\ w_4$
22bbb.	$\qquad\quad w_4 : \mathbf{T}time$
	$\qquad\quad \times$

We denote with "a" and "b" the two branches which are created by the application of β-rule to step 13., "ba" and "bb" the two ones that are created by the β-rule to step 14b., "bba" and "bbb" the two one created by the β-rule to step 17d. Explanation: *1., 2., 3., and 4.*: formula (1), (2), (3), and (4) from Example II.3.3; *5.*: goal, formula (5) from Example II.3.3; *6. and 7.*: from 5., by application of π-rule; *8. and 9.*: from 6., by π-rule; *10. and 11.*: from 7. and 9., by axiom (A_1) and ρ-rule; *12.*: from 4. and 10., by ν-rule; *13.*: from 12. and 11., by ν-rule; *14a. and 14b*: from 13, by β-rule, branch "a" closes; *15ba. and 15bb.*: from 14b., by β-rule; *16ba.*: from 3. and 10, by ν-rule; *17ba.*: from 16ba. and 11, by ν-rule, branch "ba" closes; *16bb.*: from 10., by axiom (A_6) and π-rule; *17bb.*: from 2 and 16bb., by ν-rule; *18bba. and 18bba*: from 17bb., by β-rule; *19bba.*: from 18bba. and 11., by ν-rule, branch "bba" closes; *19bbb. and 20bbb.*: from 18bbb., by π-rule; *21bbb.*: from 10. and 10bbb., by axiom (A_3) and ρ-rule; *22bbb.*: from 1. and 21bbb., by ν-rule, branch "bbb" closes.

Remark III.2.2 Note that, the ρ-rule can be regarded as a *rewriting* rule which creates new paths among worlds according to the inclusion properties of the modal logic. For instance, in Example III.2.2, in steps 10. and 11. a new path, represented by $i\ \rho_{peter}\ w_3$ and $w_3\ \rho_{john}\ w_2$, is created rewriting the path $i\ \rho_{john}\ w_1$, $w_1\ \rho_{peter}\ w_2$ (steps 7. and 9.), according to the inclusion property $\mathcal{R}_{peter} \circ \mathcal{R}_{john} \supseteq \mathcal{R}_{john} \circ \mathcal{R}_{peter}$. Moreover, the path $i\ \rho_{wife(peter)}\ w_3$ comes from $i\ \rho_{peter}\ w_3$ as well as the path $i\ \rho_{peter}\ w_4$ comes from $i\ \rho_{peter}\ w_3$, $w_3\ \rho_{peter}\ w_4$ (see Figure III.3).

Example III.2.3 *(The bungling chemist)* Assume that a chemical compound "c" is made pouring the elements "a" and, then, "b" into the same beaker. The two elements "a" and "b" are

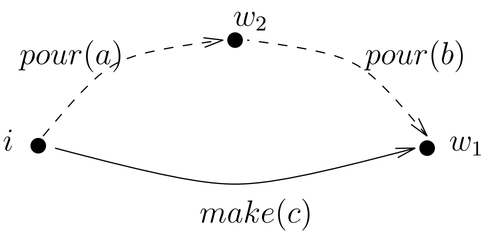

Figure III.4: ρ-rule as rewriting rule: counter-model construction of Example III.2.3.

not acid. We use the modal operator $[pour(a)]$ and $[pour(b)]$ to represent the action of pouring the element "a" and "b", respectively, and the modal operator $[make(c)]$ to denote the action of making the element "c". Thus, we have the following axiom schemas:

(A_1) $[pour(a)][pour(b)]\varphi \supset [make(c)]\varphi$;
(A_2) $[pour(b)][pour(a)]\varphi \supset [make(c)]\varphi$.

The compound "c" is not acid, unless the two different elements are not measured out carefully. Since the two elements alone are not acid, after pouring one into an empty beaker:

(1) $[pour(a)]\neg acid$

it remains not acid. Note that, however, from (1) we cannot prove the formula $\langle pour(a)\rangle\neg acid$ because the modal operator $[pour(a)]$ were not serial. Now, we add the observation that it is possible that after making the compound "c" it results acid:

(2) $\langle make(c)\rangle acid$

and so the formula $\langle pour(a)\rangle\neg acid$ is provable. Since also the formula $\langle pour(a)\rangle\langle pour(b)\rangle acid$ from (1) and (2), we can deduce that, when the compound "c" is acid, a wrong measure of element "b" with respect to the amount of element "a" already in the beaker happened. The proof is the following (see also Figure III.4):

1. $i : \mathbf{T}[pour(a)]\neg acid$
2. $i : \mathbf{T}\langle make(c)\rangle acid$
3. $i : \mathbf{F}(\langle pour(a)\rangle\neg acid \wedge \langle pour(a)\rangle\langle pour(b)\rangle acid)$
4. $w_1 : \mathbf{T}acid$
5. $i\ \rho_{make(c)}\ w_1$
6. $i\ \rho_{pour(a)}\ w_2$
7. $w_2\ \rho_{pour(b)}\ w_1$

8a. $i : \mathbf{F}\langle pour(a)\rangle\neg acid$
9a. $w_2 : \mathbf{F}\neg acid$
10a. $w_2 : \mathbf{T}\neg acid$
$\times$
8b. $i : \mathbf{F}\langle pour(a)\rangle\langle pour(b)\rangle acid$
9b. $w_2 : \mathbf{F}\langle pour(b)\rangle acid$
10b. $w_1 : \mathbf{F}acid$
$\times$

We denote with "a" and "b" the two branches which are created by the application of β-rule to step 3. Explanation: *1. and 2.*: formula (1) and (2); *3.*: goal; *4. and 5.*: from 2., by application of π-rule; *6. and 7.*: from 5., by axiom (A_1) and ρ-rule; *8a. and 8b.*: from 3.., by β-rule; *9a.*: from 8a. and 6., by ν-rule; *10a.*: from 1. and 6., by ν-rule, branch "a" closes; *9b.*: from 8b. and 6., by ν-rule; *10b.*: from 9b. and 7., by ν-rule, branch "b" closes. *15b. and 16b.*: from 14., by π-rule;

III.3 Soundness and completeness

In this section we discuss the soundness and completeness of the tableau calculus presented in the previous section. The proof follows the guideline of [Fitting, 1983, Chapter 8], and [Goré, 1995, Section 6].

Soundness

In order to prove the soundness we first prove that the tableau rules preserve the *satisfiability* but, to do this, we have to give more formally its meaning.

Let $\mathcal{L}$ be a modal language and let $\mathcal{A}$ be a set of inclusion axioms. Given a set of prefixed signed formulae and accessibility relation formulae S of $\mathcal{L}$ and a Kripke $\mathcal{A}$-interpretation $M = \langle W, \{\mathcal{R}_t \mid t \in \text{MOD}\}, V\rangle$, we say $v \in W$ is *$\mathcal{R}_t$-idealizable* if there is some $v' \in W$ such that $(v, v') \in \mathcal{R}_t$. Now, we name *$\mathcal{A}$-mapping* a mapping I from the subset of constant world symbols $\mathcal{W}_C$ that occur in some accessibility relation formula of S to W such that if $w\ \rho_t\ w' \in S$ and $I(w)$ is $\mathcal{R}_t$-idealizable then $(I(w), I(w')) \in \mathcal{R}_t$. We say S is *$\mathcal{A}$-satisfiable under the $\mathcal{A}$-mapping I in the Kripke $\mathcal{A}$-interpretation M* if, for each $w : \mathbf{T}\varphi$, $M, I(w) \models_{\mathcal{A}} \varphi$ and, for each $w : \mathbf{F}\varphi$, $M, I(w) \not\models_{\mathcal{A}} \varphi$. More generally, we call a set S of prefixed signed formulae and accessibility relation formulae *$\mathcal{A}$-satisfiable* if S is $\mathcal{A}$-satisfiable under some $\mathcal{A}$-mapping.

Therefore, a branch of a tableau is $\mathcal{A}$-satisfiable if the set of prefixed signed formulae on it is $\mathcal{A}$-satisfiable, and a tableau is $\mathcal{A}$-satisfiable if some its branch is $\mathcal{A}$-satisfiable.

Proposition III.3.1 *Let T be an $\mathcal{A}$-satisfiable prefixed tableau and let T' be the tableau which is obtained from T by means of one of the extension rules given in Figure III.2. Then, T' is also $\mathcal{A}$-satisfiable.*

Proof. The proof is made by giving an $\mathcal{A}$-mapping between prefixes which appear in a tableau and possible worlds of an appropriate Kripke $\mathcal{A}$-interpretation, whose accessibility relation respects the structure imposed by the accessibility relation formulae of the tableau. In particular, since a tableau is $\mathcal{A}$-satisfiable if one of its branches is, we can focus on application of the extension rules to that branch. The cases when the applied extension rule is either the α-rule or the β-rule are simple.

Let us assume that the branch S is $\mathcal{A}$-satisfiable under the $\mathcal{A}$-mapping I in the Kripke $\mathcal{A}$-interpretation $M = \langle W, \{\mathcal{R}_t \mid t \in \text{MOD}\}, V\rangle$ and the applied extension rule is the ν-rule to obtain S'. Let us suppose $w : \nu^t \in S$ and $S' = S \cup \{w' : \nu^t_0\}$, where w' is used on S.

Thus, $M, I(w) \models_{\mathcal{A}} \nu^t$ and I is already defined for w' and $(I(w), I(w')) \in \mathcal{R}_t$. It follows that $M, I(w') \models_{\mathcal{A}} \nu_0^t$ by definition of satisfiability relation.

The applied extension rule is the π-rule to obtain S'. Let us suppose $w : \pi^t \in S$ and $S' = S \cup \{w : \pi_0^t, w\ \rho_t\ w'\}$, where $w' \in \mathcal{W}_C$ is new on S and, therefore, I is not defined on w'. Now, $M, I(w) \models_{\mathcal{A}} \pi^t$, hence, by definition of satisfiability relation, there exists a $v \in W$ such that $(I(w), v) \in \mathcal{R}_t$ and $M, v \models_{\mathcal{A}} \pi_0^t$. This means that $I(w)$ is $\mathcal{R}_t$-idealizable and, hence, it is enough to extend the definition of I by setting $I(w') = v$.

The applied extension rule is the ρ-rule to obtain S'. Let us assume $w\ \rho_{s_1}\ w_1, \ldots, w_{m-1}\ \rho_{s_m}\ w' \in S$ and $S' = S \cup \{w\ \rho_{t_1}\ w_1', \ldots, w_{n-1}'\ \rho_{t_n}\ w'\}$, where $[t_1] \ldots [t_n]\varphi \supset [s_1] \ldots [s_m]\varphi$ is in $\mathcal{A}$ and $w_1', \ldots, w_n'$ are new on S. Then, I is already defined for $w, w_1, \ldots, w_{m-1}, w'$ and $(I(w), I(w_1)) \in \mathcal{R}_{s_1}, \ldots, (I(w_{m-1}), I(w')) \in \mathcal{R}_{s_m}$. Since M is a Kripke $\mathcal{A}$-interpretation, there exist $v_1, \ldots, v_{n-1}$ in W such that $(I(w), v_1) \in \mathcal{R}_{t_1}, \ldots, (v_{n-1}, I(w')) \in \mathcal{R}_{t_n}$. This means that $I(w)$ is $\mathcal{R}_{t_1}$-idealizable, therefore, we can extend the definition of I by setting $I(w_1') = v_1$. Now, $I(w_1')$ is $\mathcal{R}_{t_2}$-idealizable then, we can extend the definition of I by setting $I(w_2') = v_2$ and so on until $I(w_{n-1}') = v_{n-1}$. This concludes the proof. □

The soundness is stated by the following.

Theorem III.3.1 (Soundness) *Let $\mathcal{L}$ be a modal language and let $\mathcal{A}$ be a set of inclusion axiom schemas, if a formula φ of $\mathcal{L}$ is $\mathcal{T}_{\mathcal{L}}^{\mathcal{A}}$-provable then, it is $\mathcal{A}$-valid.*

Proof. By contradiction, let us assume that φ is $\mathcal{T}_{\mathcal{L}}^{\mathcal{A}}$-provable and $M, w \not\models_{\mathcal{A}} \varphi$, for some Kripke $\mathcal{A}$-interpretation $M = \langle W, \{\mathcal{R}_t \mid t \in \text{MOD}\}, V\rangle$. The tableau which starts with the formula $i : \mathbf{F}\varphi$ is $\mathcal{A}$-satisfiable by means of M by introducing an $\mathcal{A}$-mapping I and setting $I(i) = w$. By Proposition III.3.1, each possible tableau obtained from $i : \mathbf{F}\varphi$ is $\mathcal{A}$-satisfiable, but this is a contradiction because φ is $\mathcal{T}_{\mathcal{L}}^{\mathcal{A}}$-provable. □

Completeness

Before showing the completeness result we describe a *systematic tableau procedure* that produces a tableau proof if one exists and, otherwise, it produces all information necessary to construct a *counter-model.* Note that, *strong completeness* is not considered in the following.

Following [Fitting, 1983, Chapter 8], in order to deal with the prefixed signed formulae of the form $w : \nu^t$ and, in particular, to make sure $w' : \nu_0^t$ has been introduced for each constant world symbol w' such that $w\ \rho_t\ w'$ belongs to the considered branch, whenever we apply ν-rule to a prefixed signed formula of type ν^t, we add a *fresh* occurrence of it at the end of that branch. Therefore, the systematic proof procedure may consider each formula only once. To remember this it labels that formula as *finished.* Moreover, in the systematic procedure, "updating a branch with a formula" means adding the formula to end of the branch if it does not already appear on it, but doing nothing if the formula already appears on that one.

Definition III.3.1 (Systematic tableau procedure) *Let $\mathcal{L}$ be a model language and let $\mathcal{A}$ be a set of inclusion axioms. Then, a systematic attempt to produce a proof of a formula φ of $\mathcal{L}$ in the modal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ is constructed by the systematic procedure shown in Figure III.5.*

It is easy to see that the systematic procedure presented is *fair*: it considers each formula which may appear on the tableau (see [Goré, 1995, Section 6] for a similar argumentation). Hence, when we start with a formula $i : \mathbf{F}\varphi$ either it terminates and every branch on it is closed proving φ or it *must* provide an open branch which contains "enough information" to construct a counter-model to φ, that is, a Kripke interpretation in which $\neg\varphi$ is satisfiable. Note that it is possible to show the König Lemma is applicable to tableau trees generated by means of our systematic procedure, hence if the attempt to find a proof for φ fails then, an open branch must be exhibit (either finite or infinite).

The meaning of "enough information" is specified by the following definition.

Definition III.3.2 *Let $\mathcal{L}$, $\mathcal{A}$, and S be a modal language, a set of inclusion axiom schemas, and a set of prefixed signed and accessibility relation formulae in $\mathcal{L}$, respectively. Then, we say that S is $\mathcal{A}$-downward satured if:*

1. *for no atomic formula φ and no prefix w, we have $w : \mathbf{T}\varphi \in S$ and $w : \mathbf{F}\varphi \in S$;*
2. *if $w : \alpha \in S$, then $w : \alpha_1 \in S$ and $w : \alpha_2 \in S$;*
3. *if $w : \beta \in S$, then $w : \beta_1 \in S$ or $w : \beta_2 \in S$;*
4. *if $w : \nu^t \in S$, then $w' : \nu_0^t \in S$ for all w' such that $w \;\rho_t\; w' \in S$;*
5. *if $w : \pi^t \in S$, then $w' : \pi_0^t \in S$ for some w' such that $w \;\rho_t\; w' \in S$;*
6. *if $w \;\rho_{s_1}\; w_1, \dots, w_{m-1} \;\rho_{s_m}\; w' \in S$ and $[t_1]\dots[t_n]\varphi \supset [s_1]\dots[s_m]\varphi \in \mathcal{A}$, then $w \;\rho_{t_1}\; w_1', \dots, w_{n-1}' \;\rho_{t_n}\; w' \in S$, for some $w_1', \dots, w_{n-1}'$.*

Proposition III.3.2 *Let φ be a formula of $\mathcal{L}$ be a formula in the modal logic language $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ for which the systematic procedure of Figure III.5 produces an open branch S then, S is a $\mathcal{A}$-downward satured set.*

Proof. It is easy to verify that the systematic tableau procedure of Figure III.5 is closed with respect to every extension rule of the calculus. As a result we have the thesis. □

Intuitively, this proposition together with the systematic procedure play the same role of the maximal-consistent-set construction used in [Fitting, 1973]. Now, we are ready to construct our counter-model.

Definition III.3.3 (Canonical model) *Given a modal language $\mathcal{L}$, let S be a set of prefixed signed formulae and accessibility relation formulae in $\mathcal{L}$ that is $\mathcal{A}$-downward satured. The canonical model $\mathcal{M}_c^{\mathcal{A}}$ is the ordered triple $\langle W, \{\mathcal{R}_t \mid t \in \mathrm{MOD}\}, V\rangle$, where:*

```
begin
  put $i : \mathbf{F}\varphi$ at the origin;
  while the tableau is open and
    some formula is not finished do begin
      $z$ := the closest to the root and leftmost not finished formula;
      for each open branch $S$ which passes through $z$ do
        case $z$ of
          $w : \alpha$:
            update $S$ with $w : \alpha_1$ and $w : \alpha_2$;
            update $S$ with $w : \alpha_2$
          $w : \beta$:
            split the end of $S$;
            update the left fork with $w : \beta_1$;
            update the right fork with $w : \beta_2$
          $w : \nu^t$:
            for each $w\ \rho_t\ w' \in S$ do
              update $S$ with $w' : \nu_0^t$;
            add $w : \nu^t$ to the end of $S$
          $w : \pi^t$:
            choose $w'$ new on the branch $S$;
            update $S$ with $w' : \pi_0^t$;
            update $S$ with $w\ \rho_{s_i}\ w'$
          $w\ \rho_{s_i}\ w'$:
            for each $[t_1] \ldots [t_n]\varphi \supset [s_1] \ldots [s_i] \ldots [s_m]\varphi \in \mathcal{A}$ do
              for each set
                $\{w_0\ \rho_{s_1}\ w_1, \ldots, w\ \rho_{s_i}\ w', \ldots, w_{m-1}\ \rho_{s_m}\ w_m\} \subseteq S$
                such that $w_{j-1}\ \rho_{s_j}\ w_j$ precedes $w\rho_{s_i}w'$ along $S$,
                where $1 \leq j \leq m$, $(i \neq j)$, do begin
                  choose $\{w'_1, \ldots, w'_{n-1}\}$ new on the branch $S$;
                  update $S$ with $w_0\ \rho_{t_1}\ w'_1, \ldots, w'_{n-1}\ \mathcal{R}_{t_n}\ w_m$
                end
        end;
      label $z$ finished
  end
end.
```

Figure III.5: A systematic tableau procedure for propositional inclusion modal logics.

- *$W = \{w \mid w$ is used on $S\}$;*
- *for each $t \in \mathrm{MOD}$, $\mathcal{R}_t = \{(w, w') \in W \times W \mid w\ \rho_t\ w' \in S\}$;*
- *for each $p \in \mathrm{VAR}$ and each $w \in W$, we set*

$$V(w,p) = \begin{cases} \mathbf{T} & \textit{if } w : \mathbf{T}p \in S \\ \mathbf{F} & \textit{otherwise} \end{cases}$$

Proposition III.3.3 *The canonical model $\mathcal{M}_c^{\mathcal{A}}$ given by Definition III.3.3 is a Kripke $\mathcal{A}$-interpretation.*

Proof. We have to prove that each inclusion properties in $IP_{\mathcal{L}}^{\mathcal{A}}$ is satisfied by $\mathcal{M}_c^{\mathcal{A}}$. Let us suppose that $\mathcal{R}_{t_1} \circ \ldots \circ \mathcal{R}_{t_n} \supseteq \mathcal{R}_{s_1} \circ \ldots \circ \mathcal{R}_{s_m} \in IP_{\mathcal{L}}^{\mathcal{A}}$, and $(w, w') \in \mathcal{R}_{s_1} \circ \ldots \circ \mathcal{R}_{s_m}$, we have to show $(w, w') \in \mathcal{R}_{t_1} \circ \ldots \circ \mathcal{R}_{t_n}$. If $(w, w') \in \mathcal{R}_{s_1} \circ \ldots \circ \mathcal{R}_{s_m}$ then, by Definition III.3.3, there exist $w_1, \ldots, w_{m-1}$ in $\mathcal{W}_C$ such that $w\ \rho_{s_1}\ w_1, \ldots, w_{m-1}\ \rho_{s_m}\ w'$ belong to S. Now, since by hypothesis S is $\mathcal{A}$-downward satured, by point (6) of Definition III.3.2, $w\ \rho_{t_1}\ w'_1, \ldots, w'_{n-1}\ \rho_{t_n}\ w' \in S$, for some $w'_1, \ldots, w'_{n-1}$ used in S, from which our thesis. □

The following lemma states that the canonical model which is build from an open branch obtained from the systematic attempt to prove a formula φ is a counter-model of φ, that is it satisfies $\neg\varphi$ (*model existence theorem*).

Lemma III.3.1 *Given a modal language $\mathcal{L}$, if S is a set of prefixed signed formulae and accessibility relation formulae of $\mathcal{L}$ that is $\mathcal{A}$-downward satured then S is $\mathcal{A}$-satisfiable.*

Proof. Suppose S is $\mathcal{A}$-downward satured. For every formula φ and every prefix w, we have that if $w : \mathbf{T}\varphi \in S$ then $\mathcal{M}_c^{\mathcal{A}}, w \models_{\mathcal{A}} \varphi$ and if $w : \mathbf{F}\varphi \in S$ then $\mathcal{M}_c^{\mathcal{A}}, w \not\models_{\mathcal{A}} \varphi$. That is, the identity mapping $I(w) = w$ is an $\mathcal{A}$-mapping for S in the Kripke $\mathcal{A}$-interpretation $\mathcal{M}_c^{\mathcal{A}}$. The proof is by induction on the structure of φ but, for simplicity, we use the uniform notation of Smullyan already introduced. The case of formulae of type α and β are trivial. Let us suppose $w : \nu^t \in S$. Then, since S is $\mathcal{A}$-downward satured, $w' : \nu_0^t \in S$ for all w' such that $w\ \rho_t\ w' \in S$. By inductive hypothesis, we have that $\mathcal{M}_c^{\mathcal{A}}, w' \models_{\mathcal{A}} \nu_0^t$, for each world w' such that $(w, w') \in \mathcal{R}_t$ and, hence, $\mathcal{M}_c^{\mathcal{A}}, w \models_{\mathcal{A}} \nu^t$ by definition of satisfiable relation. Now, let us assume, now, $w : \pi^t \in S$. Then, since S is $\mathcal{A}$-downward satured, $w' : \pi_0^t \in S$ for some w' such that $w\ \rho_t\ w' \in S$. By inductive hypothesis, we have that $\mathcal{M}_c^{\mathcal{A}}, w' \models_{\mathcal{A}} \pi_0^t$, for some world w' such that $(w, w') \in \mathcal{R}_t$ and, hence, $\mathcal{M}_c^{\mathcal{A}}, w \models_{\mathcal{A}} \pi^t$ by definition of satisfiable relation. □

Now, we are in the position to prove the completeness of the presented tableau calculus.

Theorem III.3.2 (Completeness) *Let $\mathcal{L}$ be a modal language and let $\mathcal{A}$ be a set of inclusion axiom schemas, if a formula φ of $\mathcal{L}$ is $\mathcal{A}$-valid then, φ is $\mathcal{T}_{\mathcal{L}}^{\mathcal{A}}$-provable.*

Proof. We prove the contrapositive, by making use of the previous results. Let us assume that φ is not $\mathcal{T}_{\mathcal{L}}^{\mathcal{A}}$-provable. Then, the tableau for φ must contain some open branch S. By Proposition III.3.2, S is $\mathcal{A}$-downward satured and, therefore, we can build a Kripke $\mathcal{A}$-interpretation in which $\neg\varphi$ is satisfied by Lemma III.3.1. Thus, φ is not $\mathcal{A}$-valid. □

Chapter IV

Decidability

In the previous chapter we have defined a tableau method for the class of inclusion modal logics. The completeness result was obtained by means of a systematic tableau procedure that always finds a counter-model for a given formula if there exists one. As a result, the completeness establishes the *semi-decidability* of the inclusion modal logics. On the other hand, we wonder if this class of logics is also *decidable*, that is if it is possible to define a *decision procedure* which works for the whole class of propositional inclusion modal logics. This procedure should halt both if a counter-model exists and if a counter-model does not exist. Unfortunately, a such algorithm *does not* exist [Fariñas del Cerro and Penttonen, 1988]. Nevertheless, if more restricted classes of inclusion modal logics are considered, a decidability result can be established.

In this chapter, we show some undecidability and decidability results about inclusion modal logics. In particular, in order to show our undecidability results, we use the Fariñas del Cerro and Penttonen's technique for associating an inclusion modal logic to a formal grammar, while we use the Fischer and Ladner's *filtration method* in order to show our decidability result. It is interesting to note that our results about (un)decidability are in the line of the ones established in [Fischer and Ladner, 1979; Harel *et al.*, 1983; Harel and Paterson, 1984] for the *Propositional Dynamic Logic* [Harel, 1984; Kozen and Tiuryn, 1990].

IV.1 Grammars, languages and modal logics

In the line of [Fariñas del Cerro and Penttonen, 1988], in this section we give a method for associating with an inclusion modal logic to a formal grammar. This allows to prove some results about undecidability and decidability of inclusion modal logics.

A *grammar* is a quadruple $G = (V, T, P, S)$, where V and T are disjoint finite sets of *variables* and *terminals*, respectively. P is a finite set of *productions*, each production is of the form $\alpha \to \beta$, where the form of α and β depends on the *type* of grammar as reported in Figure IV.1. Finally, $S \in V$ is a special variable called the *start symbol* [Hopcroft and Ullman, 1979].

Class of language	*Form of production*
type-0	$\alpha \in (V \cup T)^* V (V \cup T)^*$ $\beta \in (V \cup T)^*$
type-1	$\alpha \in (V \cup T)^* V (V \cup T)^*$ $\beta \in (V \cup T)^+$ $\lvert\beta\rvert \leq \lvert\alpha\rvert$
type-2	$\alpha \in V$ $\beta \in (V \cup T)^*$
type-3	$\alpha \in V$ $\beta = \sigma A$ or $\beta = \sigma$ $\sigma \in T^*$, $A \in V$

Figure IV.1: Production grammar form for different classes of languages. We denote by "L^*" the Kleene closure of the language L (i.e. it denotes zero or more concatenation of L) and by "+" the positive closure of L (i.e. it denotes one or more concatenation of L) [Hopcroft and Ullman, 1979].

We say that the production $\alpha \rightarrow \beta$ is applied to the string $\gamma\alpha\delta$ to *directly derive* $\alpha\beta\delta$ in grammar G, written $\gamma\alpha\delta \Rightarrow_G \gamma\beta\delta$. The relation *derives*, $\Rightarrow_G^*$, is the reflexive, transitive closure of $\Rightarrow_G$. The *language generated* by a grammar G, denoted by $L(G)$ is the set of *words* $\{w \in T^* \mid S \Rightarrow_G^*\}$.

Given a tableau branch S, let w_0 and w_n two prefixes used on S, a *path* $\xi(w_0, w_n)$ is a collection $\{w_0\ \rho_{t_1}\ w_1,\ w_1\ \rho_{t_2}\ w_2,\ \ldots,\ w_{n-1}\ \rho_{t_n}\ w_n\}$ of accessibility relation formulae in S. We say that the path $\xi(w_0, w_n)$ *directly ρ-derives* the path $\xi'(w_0, w_n)$ if the path $\xi'(w_0, w_m)$ is obtained from $\xi(w_0, w_n)$ by means of the application of the ρ-rule to a sub-path of $\xi(w_0, w_n)$. The relation *ρ-derive* is the reflexive, transitive closure of the relation *directly ρ-derive.*

Example IV.1.1 Let us consider the structure of Figure III.3. Then, for instance, the path $\xi_1(i, w_2) = \{i\ \rho_{john}\ w_1,\ w_1\ \rho_{peter}\ w_2\}$ directly ρ-derives the path $\xi_2(i, w_2) = \{i\ \rho_{peter}\ w_3,$ $w_3\ \rho_{john}\ w_2\}$, the path $\xi_3(i, w_4) = \{i\ \rho_{peter}\ w_3,\ w_3\ \rho_{peter}\ w_4\}$ directly ρ-derives the path $\xi_4(i, w_4) = \{i\ \rho_{wife(peter)}\ w_3,\ w_3\ \rho_{peter}\ w_4\}$, and the path $\xi_1(i, w_2)$ ρ-derives the path $\xi_5(i, w_2) =$ $i\ \rho_{wife(peter)}\ w_3,\ w_3\ \rho_{john}\ w_2$.

Due to the similarity between inclusion modal axioms and the production rules in a grammar, we can associate to a given grammar a corresponding inclusion modal logic. More precisely, following [Fariñas del Cerro and Penttonen, 1988], given a formal grammar $G = (V, T, P, S)$, we define an inclusion modal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ *based on* G as follows:

- the set MOD is $(V \cup T)$;
- the set $\mathcal{A}$ of inclusion axioms contains a schema $[t_1] \ldots [t_n]\varphi \supset [s_1] \ldots [s_m]\varphi$ for each production $t_1 \cdots t_n \rightarrow s_1 \cdots s_m \in P$.

We call *unrestricted*, *context sensitive*, *context-free*, and *right-regular* modal logic an inclusion modal logic based on a type-0, type-1, type-2, and type-3 grammar, respectively.

Example IV.1.2 Consider, for instance, the grammar G, where:

- $V = \{A\}$;
- $T = \{b\}$;
- $P = \{A \to \varepsilon, A \to A\ A, A \to b\ A\}$;
- $S = A$.

Then, the inclusion modal logic $\mathcal{I}^A_{\mathcal{L}}$ based on G contains the inclusion axioms:

- $[A]\varphi \supset \varphi$,
- $[A]\varphi \supset [A][A]\varphi$, and
- $[A]\varphi \supset [b][A]\varphi$

(i.e., $\mathcal{I}^A_{\mathcal{L}}$ is axiomatized by $KT4(A) + 4M(A, b)$).

Remark IV.1.1 Note that, the class of unrestricted inclusion modal logics is equivalent to the class of inclusion modal logics.

If $\xi(w_0, w_n)$ is the path $\{w_0\ \rho_{t_1}\ w_1,\ \dots,\ w_{n-1}\ \rho_{t_n}\ w_n\}$, we denote by $\overline{\xi}(w_0, w_n)$ the sequence of labels $t_1 \cdots t_n$ (called *word*). It is easy to verify the following proposition.

Proposition IV.1.1 *If $\xi(w_0, w_n)$ is a path in a tableau branch starting from a formula of an inclusion modal logic $\mathcal{I}^A_{\mathcal{L}}$ based on a grammar G then, $\xi(w_0, w_n)$ ρ-derives a path $\xi'(w_0, w_n)$ if and only if $\overline{\xi'}(w_0, w_n) \Rightarrow^*_G \overline{\xi}(w_0, w_n)$.*

An interesting case (that will be used later on) is the following. Consider the type-3 grammar $G = (\{S\}, T, P, S)$, where the set P contains the productions $S \to t$ and $S \to S\ t$ for each $t \in T$, then $L(G) = T^*$. Let $\mathcal{I}^A_{\mathcal{L}}$ be the inclusion modal logic based on G and let us consider the formula

$$\varphi_T(q) = \bigwedge_{t \in T} (\langle t \rangle q \wedge [S]\langle t \rangle q)$$

where $q \in$ VAR. Then, a tableau starting from $i : \mathbf{T}\varphi_T(q)$ is formed by only one branch that goes on forever. The interesting is that for each word $x \in T^*$ the tableau branch contains a path $\xi(i, w)$ such that $\overline{\xi}(i, w) = x$ (see Figure IV.2).

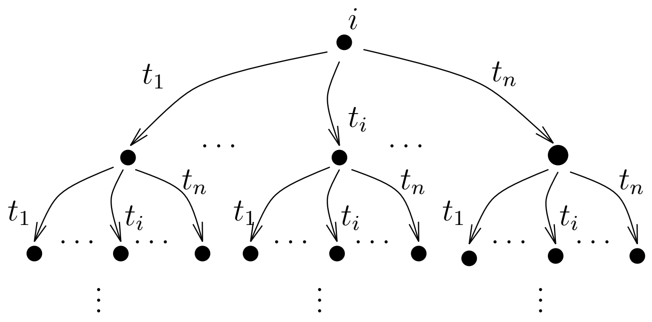

Figure IV.2: The Kripke structure generated by proving $\varphi_T(q)$.

IV.2 Undecidability results for inclusion modal logics

The tableau method developed in the previous chapter allows to generalize the Fariñas del Cerro and Penttonen's observations about the correspondence between the membership problem and the validity problem of inclusion logics as stated by the following theorem.

Theorem IV.2.1 *Given a grammar $G = (V, T, P, S)$, let $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ be the inclusion modal logic based on G. Then, for any propositional variable p of $\mathcal{L}$, $\models_{\mathcal{A}} [S]p \supset [s_1] \dots [s_m]p$ if and only if $S \Rightarrow_G^* s_1 \cdots s_m$, where the s_i's are in $V \cup T$.*

Proof. (*If* part) Let us suppose that $\models_{\mathcal{A}} [S]p \supset [s_1] \dots [s_m]p$, then, the tableau starting from:

1.	$i : \mathbf{F}([S]p \supset [s_1] \dots [s_m]p)$

closes by Theorem III.3.2. Now, by applying the β-rule we obtain:

2.	$i : \mathbf{T}[S]p$
3.	$i : \mathbf{F}[s_1] \dots [s_m]p$

and applying m times the π-rule:

4.	$w_1 : \mathbf{F}[s_2] \dots [s_m]p$
5.	$i \ \rho_{s_1} \ w_1$
$\dots$	$\dots$
$2m + 3.$	$w_m : \mathbf{F}p$
$2m + 4.$	$w_{m-1} \ \rho_{s_m} \ w_m$

Since, by hypothesis, the above tableau closes, the only way for this to happen is that after a finite number of applications of the ρ-rule we have the prefixed signed formula $w_m : \mathbf{T}p$ in the branch. This happens if the path $\xi(i, w_m) = \{i \ \rho_{s_1} \ w_1, \ \dots, \ w_{m-1} \ \rho_{s_m} \ w_m\}$ ρ-derives the path $\xi'(i, w_m) = \{i \ \rho_S \ w_m\}$, that is, if there exits a derivation $\overline{\xi'}(i, w_m) = S \Rightarrow_G^* \overline{\xi}(i, w_m) = s_1 \cdots s_m$ by Proposition IV.1.1. (*Only if* part) Assume that there exists a derivation $S \Rightarrow_G^* s_1 \cdots s_m$. Since a systematic attempt to prove the formula $i : \mathbf{F}([S]p \supset [s_1] \ldots [s_m]p)$ generates a path $\xi(i, w_m) = \{i \ \rho_{s_1} \ w_1, \ \ldots, \ w_{m-1} \ \rho_{s_m} \ w_m\}$ and $\xi(i, w_m)$ ρ-derives the path $\xi'(i, w_m) = \{i \ \rho_S \ w_m\}$, after a finite number of steps, the only branch of the tableau closes by $w_m : \mathbf{T}p$ and $w_m : \mathbf{F}p$. $\Box$

Thus, taking into account that it is undecidable to establish if a word belongs to the language generated by an arbitrary type-0 grammar [Hopcroft and Ullman, 1979], we have the following corollary.

Corollary IV.2.1 *The validity problem for the class of inclusion modal logics is undecidable.*

Indeed, this result has already been shown in [Fariñas del Cerro and Penttonen, 1988]. However, Fariñas del Cerro and Penttonen were not able to prove Theorem IV.2.1 for the modal logics based on type-0 grammars. This is why they focused on a subclass of the inclusion modal logics, that they call *Thue logics*, proving the undecidability of inclusion modal logics by showing that the Thue logics are undecidable. A Thue logic is an inclusion modal logic based on a *Thue system* [Book, 1987], that is a type-0 grammar whose productions are *symmetric*. Thus, the Thue logics are inclusion modal logics characterized by axiom schemas where the implication is replaced by the biimplication. Since the word problem for the Thue systems is proved *undecidable* (see [Book, 1987]), proving that a formula is a theorem of a Thue logic will be undecidable.[1]

In [Fariñas del Cerro and Penttonen, 1988] some problems are left open. We wonder if more restricted classes of logics (e.g. modal logics based on context sensitive, context-free, regular grammars) are decidable. In the following, we show that also the class of context sensitive and context-free inclusion modal logics are undecidable by reducing the solvability of the problem $L_1 \cap L_2 \neq \emptyset$ (where L_1 and L_2 are languages generated by either type-1 or type-2 grammars) to the satisfiability of formulas of context sensitive and context-free inclusion modal logics.

Theorem IV.2.2 *Let $G_1 = (V_1, T_1, P_1, S_1)$ and $G_2 = (V_2, T_2, P_2, S_2)$ be two grammars such that $V_1 \cap V_2 = \emptyset$ and $T_1 = T_2 \neq \emptyset$. Then, there exists an inclusion modal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ and a formula φ of $\mathcal{L}$ such that $\models_{\mathcal{A}} \varphi$ if and only if $L(G_1) \cap L(G_2) \neq \emptyset$.*

Proof. Let us define the grammar $G = (V, T, P, S)$, where:

- $V = V_1 \cup V_2 \cup \{S\}$;
- $T = T_1 = T_2$;
- $P = P_1 \cup P_2 \cup \{S \rightarrow t, S \rightarrow S \ t \mid t \in T\}$;

[1]The Thue systems have also been used in [Kranсht, 1995] to define logics similar to those studied in [Fariñas del Cerro and Penttonen, 1988], which, however, are not in the class of inclusion modal logics because modal operators enjoy some further properties like seriality and determinism. In [Krancht, 1995] undecidability results are proved for this class of logics.

- $S \notin V_1$ and $S \notin V_2$.

Then, we assume as $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}}$ the inclusion modal logic based on G and

$$\varphi = \varphi_T(q) \supset ([S_1]p \supset \langle S_2\rangle p)$$

where $p, q \in$ VAR and $p \neq q$. (*If* part) Suppose that $\models_{\mathcal{A}} \varphi$ then, the tableau starting from:

1. $i : \mathbf{F}(\varphi_T(q) \supset ([S_1]p \supset \langle S_2\rangle p))$

must close. Now, by applying twice the β-rule we obtain:

2. $i : \mathbf{T}\varphi_T(q)$
3. $i : \mathbf{T}[S_1]p$
4. $i : \mathbf{F}\langle S_2\rangle p$

Since, by hypothesis, the above tableau closes, the only way for this to happen is that after a finite number of steps we must have a prefixed signed formula $w : \mathbf{T}p$ and a prefixed signed formula $w : \mathbf{F}p$ for some prefix w and, therefore, a path $\xi(i, w)$ that ρ-derives both the path $\xi_1(i, w) = \{i \ \rho_{S_1} \ w\}$ and the path $\xi_2(i, w) = \{i \ \rho_{S_2} \ w\}$. Thus, there is both a derivation $\overline{\xi_1}(i, w) = S_1 \Rightarrow^*_G \overline{\xi}(i, w)$ and a derivation $\overline{\xi_1}(i, w) = S_2 \Rightarrow^*_G \overline{\xi}(i, w)$ by Proposition IV.1.1, i.e, $\overline{\xi}(i, w) \in L(G_1)$ and $\overline{\xi}(i, w) \in L(G_2)$ ($S_1 \Rightarrow^*_G \overline{\xi}(i, w)$ and $(S_2 \Rightarrow^*_G \overline{\xi}(i, w))$, i.e. $\overline{\xi}(i, w) \in L(G_1) \cap L(G_2)$. (*Only if* part) Assume that $L(G_1) \cap L(G_2) \neq \emptyset$ then, there exists a word $x \in T^*$ such that $x \in L(G_1)$, that is $S_1 \Rightarrow^*_{G_1} x$, and $x \in L(G_2)$, that is $S_2 \Rightarrow^*_{G_2} x$. Since a systematic attempt to prove the formula $i : \mathbf{T}\varphi_T(q)$ can generate a path $\xi(i, w)$, for some prefix w, such that $\overline{\xi}(i, w) = y$, for any $y \in T^*$, after a finite number of steps we have a path $\xi'(i, w')$ such that $\overline{\xi'}(i, w') = x$. Thus, we have also a path $\xi'_1(i, w') = \{i \ \rho_{S_1} \ w'\}$ and a path $\xi'_2(i, w') = \{i \ \rho_{S_2} \ w'\}$ by application of a finite number of the ρ-rule. This is enough to close the only branch of the tableau by $w' : \mathbf{T}p$ and $w' : \mathbf{F}p$. $\Box$

Thus, taking into account that if G_1 and G_2 are two arbitrary type-1 (type-2) grammars then it is undecidable if $L(G_1) \cap L(G_2) \neq \emptyset$ [Hopcroft and Ullman, 1979], we have the following corollary.

Corollary IV.2.2 *The validity problem for the class of context-free inclusion modal logic is undecidable.*

Remark IV.2.1 Since the problem if $L_1 \cap L_2 \neq \emptyset$ is undecidable also for the class of *deterministic* type-2 grammars, the validity problem for the inclusion modal logics based on this kind of grammars is undecidable.

IV.3 A decidability result for inclusion modal logics

In the previous section we have shown that it is not possible to supply a general decision procedure for the class of inclusion modal logics based on unrestricted, context sensitive and context-free grammars. In this section, instead, we give a *decidability* result for the inclusion modal logics based on *right* type-3 formal grammars, that is, those ones based on grammars whose productions are of the form $A \to \sigma$ or $A \to \sigma\ A'$, where A and A' are variables and σ a string of terminals. In order to do this, we modify the *filtration method* for dynamic logic extending the definition of Fisher-Ladner closure [Fischer and Ladner, 1979].

Remark IV.3.1 Let $G = (V, T, P, S)$ be a right type-3 grammar and let A be a variable. Then, every *sentential form* derived from A has the form σX, where $\sigma \in T^*$ and either $X \in T$ or $X \in V$.

Definition IV.3.1 *Let $G = (V, T, P, S)$ be a right type-3 grammar and let A be a variable. Then, a derivation of a sentential form σX from A is said to be* non-recursive *if and only if each variable of V appears in the derivation, apart from σX, at most once.*

Some useful properties about non-recursive derivations of right type-3 grammars are the following.

Proposition IV.3.1 *Let $G = (V, T, P, S)$ be a right type-3 grammar, let A_0 be a variable and let $A_0 \Rightarrow_G^* \sigma_1 \cdots \sigma_n A_n \Rightarrow_G \sigma_1 \cdots \sigma_n \sigma_{n+1} A_{n+1}$ be a derivation, where either $A_{n+1} \in V$ or $A_{n+1} \in T$ and $A_i \to \sigma_{i+1} A_{i+1} \in P$, for $i = 0, \ldots, n$. Then, there exists a non-recursive derivation $A_0 \Rightarrow_G^* \sigma_1 \ldots \sigma_i \sigma_{n+1} A_{n+1}$, $0 \leq i \leq n$.*

Proof. If the derivation $A_0 \Rightarrow_G^* \sigma_1 \cdots \sigma_n A_n \Rightarrow_G \sigma_1 \cdots \sigma_n \sigma_{n+1} A_{n+1}$ is not non-recursive then, there are A_i and A_j, with $0 \leq i < j \leq n$, such that $A_i = A_j$. That is, $A_0 \Rightarrow_G^* \sigma_1 \cdots \sigma_i A_i \Rightarrow_G^* \sigma_1 \cdots \sigma_i \cdots \sigma_j A_j \Rightarrow_G^* \sigma_1 \cdots \sigma_n \sigma_{n+1} A_{n+1}$. Thus, there exists a derivation $A_j \Rightarrow_G^* \sigma_{j+1} \cdots \sigma_{n+1} A_{n+1}$ and, therefore, a derivation $A_0 \Rightarrow_G^* \sigma_1 \cdots \sigma_i A_j \Rightarrow_G^* \sigma_1 \cdots \sigma_i \sigma_{j+1} \cdots \sigma_{n+1} A_{n+1}$. Now, if this derivation is non-recursive we have our thesis otherwise we repeat the above transformation on the new derivation just obtained. Now, the number of variables that appear on the original derivation is finite and it decreases at any stage of the transformation, moreover, the cardinality of the set of variables is also finite. Thus, the process always terminates leading to a non-recursive derivation. □

Proposition IV.3.2 *Let $G = (V, T, P, S)$ be a right type-3 grammar. Then, the number of non-recursive derivations that start with a variable of G is bounded.*

Proof. The maximum length (number of directly derivation steps) of a non-recursive derivation is equal to the cardinality $|V|$ of the set of variables V. Then, let n be the maximum number of productions associated to a variable of V, a bound of the number of different non-recursive derivations starting from a fixed variables is $\sum_{i=1}^{|V|} n^i$. Therefore, the number of different non-recursive derivations that start with a variable of G is bounded by $\mathrm{der}_G = |V| \cdot \sum_{i=1}^{|V|} n^i$. □

Let $G = (V, T, P, S)$ be a right type-3 grammar and $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}}$ the regular inclusion modal logic based on G. Then, we define the *Fischer-Ladner closure* $FL(\varphi)$ of a formula φ of $\mathcal{L}$ (that uses only *existential modal operators*, *or*, and *negation*[2]) as follows:

- if $\psi \vee \psi' \in FL(\varphi)$ then $\psi \in FL(\varphi)$ and $\psi' \in FL(\varphi)$;
- if $\neg\psi \in FL(\varphi)$ then $\psi \in FL(\varphi)$;
- if $\langle t\rangle\psi \in FL(\varphi)$ and $t \in T$ then $\psi \in FL(\varphi)$;
- if $\langle A\rangle\psi \in FL(\varphi)$, $A \in V$, and there is a non-recursive derivation $A \Rightarrow^*_G t_1 \cdots t_n X$, where $t_1, \ldots, t_n \in T$ and either $X \in T$ or $X \in V$, then $\langle t_1\rangle \ldots \langle t_n\rangle\langle X\rangle\psi \in FL(\varphi)$.

It is worth noting that the Fischer-Ladner closure is finite for any formula of a right regular inclusion modal logic because the number of non-recursive derivations is finite if the length of the formula φ is finite. In particular, let $|\varphi|$ be the length (number of symbols) of φ then, $|FL(\varphi)| \leq |\varphi| \cdot m \cdot |V| \cdot \mathrm{der}_G$, where m is the maximum length of a production of the grammar.[3]

Let $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}}$ be the inclusion modal logic based on a type-3 grammar $G = (V, T, P, S)$ and consider a Kripke $\mathcal{A}$-interpretation $M = \langle W, \{R_t \mid t \in \mathrm{MOD}\}, V\rangle$ and a formula φ of $\mathcal{L}$. Then, we define an *equivalence* relation $\equiv$ on state of W by

$$w \equiv w' \text{ iff } \forall\psi \in FL(\varphi), M, w \models_{\mathcal{A}} \psi \Leftrightarrow M, w' \models_{\mathcal{A}} \psi$$

we use the notation $\overline{w}$ for this equivalence class. The *quotient* Kripke $\mathcal{A}$-interpretation

$$M^{FL(\varphi)} = \langle W^{FL(\varphi)}, \{\mathcal{R}^{FL(\varphi)}_t \mid t \in \mathrm{MOD}\}, V^{FL(\varphi)}\rangle$$

(the *filtration of M through $FL(\varphi)$*) is defined as follows:

- $W^{FL(\varphi)} = \{\overline{w} \mid w \in W\}$;
- $V^{FL(\varphi)}(\overline{w}, p) = V(w, p)$, for any $p \in \mathrm{VAR}$ and $\overline{w} \in W^{FL(\varphi)}$;
- $\mathcal{R}^{FL(\varphi)}_t \supseteq \{(\overline{w}, \overline{w'}) \in W^{FL(\varphi)} \times W^{FL(\varphi)} \mid (w, w') \in \mathcal{R}_t\}$.

  Moreover, $\mathcal{R}^{FL(\varphi)}_t$ is closed with respect to the inclusion axioms, that is, for each inclusion axiom schema $[t]\alpha \supset [s_1] \ldots [s_m]\alpha$ if $(\overline{w_0}, \overline{w_1}) \in \mathcal{R}^{FL(\varphi)}_{s_1}$, ..., $(\overline{w_{m-1}}, \overline{w_m}) \in \mathcal{R}^{FL(\varphi)}_{s_m}$ then the pair $(\overline{w_0}, \overline{w_m})$ belongs to the accessibility relation $\mathcal{R}^{FL(\varphi)}_t$.

The following lemma states that when we insert any extra binary relation between $\overline{w}$ and $\overline{w'}$ in a accessibility relation $\mathcal{R}^{FL(\varphi)}_t$ of $M^{FL(\varphi)}$, in order to satisfy the relative set of inclusion properties $IP^{\mathcal{A}}_{\mathcal{L}}$, it is not the case that there was any $\langle t\rangle\psi \in FL(\varphi)$ which was true at w while ψ itself was false at w' (see [Hughes and Cresswell, 1984, page 137]).

[2]Since all other connectives can be defined in terms of these, this is not a restrictive condition.

[3]In fact, each subformulae of φ could be introduced in $FL(\varphi)$. Every subformulae with associated every possible sequence of modalities that comes from a non-recursive derivation whose length is at the maximum m times $|V|$.

Lemma IV.3.1 *For all $\psi = \langle t\rangle\psi' \in FL(\varphi)$, if $(\overline{w}, \overline{w'}) \in \mathcal{R}_t^{FL(\varphi)}$ and $M, w' \models_{\mathcal{A}} \psi'$ then $M, w \models_{\mathcal{A}} \langle t\rangle\psi'$.*

Proof. Assume that $\psi = \langle t\rangle\psi' \in FL(\varphi)$ then, $\psi' \in FL(\varphi)$ by definition of Fischer-Ladner closure. Now, there are two cases, which depend on whether $(\overline{w}, \overline{w'}) \in \mathcal{R}_t^{FL(\varphi)}$ has been added to initial definition of filtration because an inclusion axiom schema of the form $[t]\alpha \supset [s_1] \dots [s_m]\alpha \in \mathcal{A}$ or not.

Assume that $(\overline{w}, \overline{w'}) \in \mathcal{R}_t^{FL(\varphi)}$ has not been added. Then, by definition of $\mathcal{R}_t^{FL(\varphi)}$, there exist w_1, $w_1' \in W$ such that $(w_1, w_1') \in \mathcal{R}_t$, $w_1 \equiv w$, and $w_1' \equiv w'$. Since $M, w' \models_{\mathcal{A}} \psi'$, $M, w_1' \models_{\mathcal{A}} \psi'$ because $\psi' \in FL(\varphi)$ and $w' \equiv w_1'$. Hence, $M, w_1 \models_{\mathcal{A}} \langle t\rangle\psi'$ because $(w_1, w_1') \in \mathcal{R}_t$. Finally, $M, w \models_{\mathcal{A}} \langle t\rangle\psi'$, because $\langle t\rangle\psi' \in FL(\varphi)$ and $w \equiv w'$.

Assume that $(\overline{w}, \overline{w'}) \in \mathcal{R}_t^{FL(\varphi)}$ but $(w, w') \notin \mathcal{R}_t$. The pair $(\overline{w}, \overline{w'})$ has been added in $\mathcal{R}_t^{FL(\varphi)}$ by the closure operation in order to satisfy an inclusion property of an inclusion axiom of the form $[t]\alpha \supset [s_1] \dots [s_m]\alpha \in \mathcal{A}$. Then, there exist $\overline{w_1}, \dots, \overline{w_{m-1}}$ such that $(\overline{w_0}, \overline{w_1}) \in \mathcal{R}_{s_1}^{FL(\varphi)}, \dots, (\overline{w_{m-1}}, \overline{w_m}) \in \mathcal{R}_{s_m}^{FL(\varphi)}$, where w_0 is w and w_m is w'. Now, in turn, for each pair $(\overline{w_{i-1}}, \overline{w_i}) \in \mathcal{R}_{s_i}^{FL(\varphi)}$, for $i = 1, \dots, n$, either $(\overline{w_{i-1}}, \overline{w_i})$ has been added by the closure operation or not. Going on this way, we have $(\overline{v_0}, \overline{v_1}) \in \mathcal{R}_{t_1}^{FL(\varphi)}, \dots, (\overline{v_{h-1}}, \overline{v_h}) \in \mathcal{R}_{t_h}^{FL(\varphi)}$ such that the corresponding pairs belong $\mathcal{R}_t$ and $t \Rightarrow_G^* t_1 \cdots t_h$, v_0 is w_0 (that, in turn, is w), and v_h is w_m (that, in turn, is w'). By construction, there exist $v_{i-1}', v_i'' \in W$ such that $(v_{i-1}', v_i'') \in \mathcal{R}_{t_i}^{FL(\varphi)}$ and $v_{i-1} \equiv v_{i-1}'$ and $v_i \equiv v_i''$, for $i = 1, \dots, h$.

Assume that $t \Rightarrow_G^* t_1 \cdots t_h$ is the derivation $A_0 \Rightarrow_G \sigma_1 A_1 \Rightarrow_G \dots \Rightarrow_G \sigma_1 \cdots \sigma_n A_n \Rightarrow_G \sigma_1 \cdots \sigma_n \sigma_{n+1}$, where A_0 is t and $A_n \rightarrow \sigma_{n+1}$ and $A_{i-1} \rightarrow \sigma_i A_i$, for $i = 1, \dots, n$, are in P, and that σ_{n+1} is $d_1 \cdots d_r$ ($= t_{h-r+1} \cdots t_h$). We know that $M, v_h \models_{\mathcal{A}} \psi'$ and we have to prove that $M, v_{h-r+1} \models_{\mathcal{A}} \langle d_1\rangle \dots \langle d_r\rangle\psi'$. Assuming that $\langle d_1\rangle \dots \langle d_r\rangle\psi' \in FL(\varphi)$ then, we have that $M, v_h'' \models_{\mathcal{A}} \psi'$ since $v_h \equiv v_h''$ and $\psi' \in FL(\varphi)$. Since $(v_{h-1}', v_h'') \in \mathcal{R}_{t_h}$ and $M, v_h'' \models_{\mathcal{A}} \psi'$ then, $M, v_{h-1}' \models_{\mathcal{A}} \langle d_r\rangle\psi'$ and, since $\langle d_r\rangle\psi' \in FL(\varphi)$ and $v_{h-1}' \equiv v_{h-1}''$, we have that $M, v_{h-1}'' \models_{\mathcal{A}} \langle d_r\rangle\psi'$. We can proceed so on until $M, v_{h-r+1}'' \models_{\mathcal{A}} \langle d_1\rangle \dots \langle d_r\rangle\psi'$ and $M, v_{h-r+1} \models_{\mathcal{A}} \langle d_1\rangle \dots \langle d_r\rangle\psi'$ since $v_{h-r+1} \equiv v_{h-r+1}''$. Now, since the inclusion axiom $[A_n]\alpha \supset [d_1] \dots [d_r]\alpha$ belongs to $\mathcal{A}$, $M, v_{h-r+1} \models_{\mathcal{A}} \langle A_n\rangle\psi'$. We can repeat the above argumentation for all derivation steps from A_0 obtaining at the end our thesis $M, w \models_{\mathcal{A}} \langle A_0\rangle\psi'$, that is, $M, w \models_{\mathcal{A}} \langle t\rangle\psi'$.

We have now to prove that $\langle d_1\rangle \dots \langle d_r\rangle\psi' \in FL(\varphi)$. By hypothesis $\langle A_0\rangle\psi' \in FL(\varphi)$ (A_0 is t) and $A_0 \Rightarrow_G^* \sigma_1 \cdots \sigma_n \sigma_{n+1}$. Then, by Proposition IV.3.1, there exists a non-recursive derivation $A_0 \Rightarrow_G^* \sigma\sigma_{n+1}$, for some $\sigma \in T^*$. By definition of Fischer-Ladner closure, since $\langle A_0\rangle\psi' \in FL(\varphi)$, we have $\langle t_1'\rangle \dots \langle t_{n'}'\rangle\langle d_1\rangle \dots \langle d_r\rangle\psi' \in FL(\varphi)$, where σ is $t_1' \cdots t_{n'}'$ and σ_{n+1} is $d_1 \cdots d_r$, and, hence, $\langle d_1\rangle \dots \langle d_r\rangle\psi'$. $\square$

Lemma IV.3.2 (Filtration Lemma) *For all $\psi \in FL(\varphi)$,*

$$M, w \models_{\mathcal{A}} \psi \textit{ if and only if } M^{FL(\varphi)}, \overline{w} \models_{\mathcal{A}} \psi.$$

Proof. The proof is by induction on the structure of ψ. (*Base* step) For $\psi \in$ VAR the thesis holds trivially. (*Induction* step) The cases $\psi = \psi' \vee \psi''$ and $\psi = \neg\psi'$ are immediate from the definitions. Assume that $\psi = \langle t \rangle \psi'$. (*If* part) If $M, w \models_{\mathcal{A}} \langle t \rangle \psi'$ then there exists w' such that $M, w' \models_{\mathcal{A}} \psi'$ and $(w, w') \in \mathcal{R}_t$. By definition, we have $(\overline{w}, \overline{w'}) \in \mathcal{R}_t^{FL(\varphi)}$ and, by induction hypothesis, $M^{FL(\varphi)}, \overline{w'} \models_{\mathcal{A}} \psi'$. Hence $M^{FL(\varphi)}, \overline{w} \models_{\mathcal{A}} \langle t \rangle \psi'$. (*Only if* part) If $M^{FL(\varphi)}, \overline{w} \models_{\mathcal{A}} \langle t \rangle \psi'$ then, there exists $\overline{w'} \in W^{FL(\varphi)}$ such that $M^{FL(\varphi)}, \overline{w'} \models_{\mathcal{A}} \psi'$ and $(\overline{w}, \overline{w'}) \in \mathcal{R}_t^{FL(\varphi)}$. By inductive hypothesis, we have that $M, w' \models_{\mathcal{A}} \psi'$ and, by Lemma IV.3.1 since $(\overline{w}, \overline{w'}) \in \mathcal{R}_t^{FL(\varphi)}$, $M, w \models_{\mathcal{A}} \langle t \rangle \psi'$. $\square$

Theorem IV.3.1 (Small Model Theorem) *Let φ be a satisfiable formula of an inclusion modal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ based on a type-3 grammar G. Then, φ is satisfied in a Kripke $\mathcal{A}$-interpretation with no more that $2^{|FL(\varphi)|}$ states.*

Proof. If φ is satisfiable, then there is a Kripke $\mathcal{A}$-interpretation M and a state w in M such that $M, w \models_{\mathcal{A}} \varphi$. Let $FL(\varphi)$ be the Fischer-Ladner closure of φ. By Lemma IV.3.2, $M^{FL(\varphi)}, \overline{w} \models_{\mathcal{A}} \varphi$. Moreover, since $|FL(\varphi)|$ is bounded by Proposition IV.3.2, then the filtration through $FL(\varphi)$ is a Kripke interpretation having at most $2^{|FL(\varphi)|}$ worlds (equivalence classes of worlds in the initial model), that being the maximum number of ways that worlds can disagree on sentences in $FL(\varphi)$. $\square$

Remark IV.3.2 A modal logic is decidable if it has the *finite model property* (i.e., if and only if each non-theorem of the modal logic is false in some finite Kripke interpretation of the logic) and it is axiomatizable by a finite number of axiom schemas. In fact, in this case there is both a positive and negative test for theorem-hood in the logic. The positive test is given by generating all the proofs of theorems in some definite order (this is possible because the axiomatization is finite, in our case also by the completeness of the tableau calculus), while for the negative test we can give a complete enumeration of the finite Kripke interpretations (models) since each Kripke interpretation is finite. Then, if a formula is a non-theorem of the logic it is false in some finite Kripke interpretation and to find this one we can examine each Kripke interpretation of the logic (a finite task since the Kripke interpretation is finite and the logic is finitely axiomatized) checking if the selected Kripke interpretation falsify the formula (a finite task since the model is finite) [Hughes and Cresswell, 1984; Chellas, 1980].

As a corollary, since each inclusion modal logic based on a right regular grammar is axiomatizable by a finite number of axiom schemas and, by Theorem IV.3.1, it is determined by a class of finite standard Kripke interpretations and, hence, it has the *finite model property* (see [Hughes and Cresswell, 1984, Chapter 8] and [Chellas, 1980, Chapter 5]), we have the following corollary.

Corollary IV.3.1 *The validity problem for the class of right-regular inclusion modal logics is decidable.*

As a final remark, it is worth noting that the systematic procedure given in the previous chapter is not a *decision procedure*: it goes on forever also when it deals with a decidable logic.

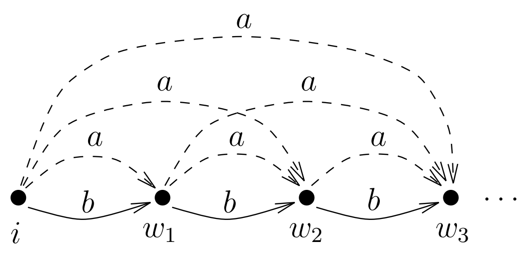

Figure IV.3: Non-terminating Kripke $\mathcal{A}$-interpretation construction of Example IV.3.1.

Example IV.3.1 Let us consider the modal logic whose set $\mathcal{A}$ of inclusion axioms consists of:

(A_1)	$[a]\varphi \supset [b][a]\varphi$
(A_2)	$[a]\varphi \supset [b]\varphi$

Despite the fact $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}}$ is decidable (it belongs to the class of right regular inclusion modal logics), the systematic attempt to prove the formula $\langle b\rangle p \supset \langle a\rangle[b]p$ runs forever (see also Figure IV.3):

1.	$i : \mathbf{F}(\langle b\rangle p \supset \langle a\rangle[b]p)$
2.	$i : \mathbf{T}\langle b\rangle p$
3.	$i : \mathbf{F}\langle a\rangle[b]p$
4.	$w_1 : \mathbf{T}p$
5.	$i\ \rho_b\ w_1$
6.	$i\ \rho_a\ w_1$
7.	$w_1 : \mathbf{F}[b]p$
8.	$w_2 : \mathbf{F}p$
9.	$w_1\ \rho_b\ w_2$
10.	$w_1\ \rho_a\ w_2$
10.	$i\ \rho_a\ w_2$
11.	$w_2 : \mathbf{F}[b]p$
12.	$w_3 : \mathbf{F}p$
13.	$w_2\ \rho_b\ w_3$
14.	$w_2\ \rho_a\ w_3$
15.	$w_1\ \rho_a\ w_3$
16.	$i\ \rho_a\ w_3$
17.	$w_3 : \mathbf{F}[b]p$
. . .	. . .

There is no hope to close the branch continuing the computation: an infinite sequence of worlds is introduced.

Chapter V

First-Order

In this chapter we extend the *propositional* modal languages in order to deal with the *predicative* case. First of all, we introduce the syntax and, then, the possible-worlds semantics. With regard to model theory, we associate with each possible world a *domain of individuals* and we have chosen to impose a *monotonicity condition* on them with respect to the accessibility relations. Afterwards, we update the tableau calculus presented in Chapter III in order to deal with quantifiers.

V.1 Syntax

The *alphabet* of a *first-order multimodal language* $\mathcal{L}_{FO}$ contains:

- a countable set VAR of *individual variables* (*variable* for short);
- for each $n \geq 0$, a countable set FUNC^n of n-place *function symbols*;
- for each $n \geq 0$, a nonempty countable set PRED^n of n-place *predicate symbols*;
- the *classical connectives* " $\wedge$ " (*and*), "$\vee$" (*or*), "$\neg$" (*not*), " $\supset$ " (*implies*);
- the *universal quantifier* "$\forall$" and *existential quantifier* "$\exists$";
- a *modal operator constructor* "[.]";
- left and right *parentheses* "(", ")", and a *comma* ",".

The set TERM of *terms* is defined to be the least set that satisfies the following conditions:

- VAR $\subseteq$ TERM;
- if $t_1, \ldots, t_n \in$ TERM and $f \in$ FUNC^n then $f(t_1, \ldots, t_n) \in$ TERM.

A 0-place function symbol is a *constant symbol*; the term $c()$ is written as c. We will assume that $\mathcal{L}_{FO}$ contains at least one constant symbol. A term is a *ground* if it does not contain any variable.

The set FOR of *formulae* of a modal language $\mathcal{L}_{FO}$ is defined to be the least set that satisfies the following conditions:

- if $t_1, \ldots, t_n \in$ TERM and $p \in \mathrm{PRED}^n$ then $p(t_1, \ldots, t_n) \in$ FOR;
- if $\varphi, \psi \in$ FOR then $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \supset \psi) \in$ FOR;
- if $x \in$ VAR and $\varphi \in$ FOR then $((\forall x)\varphi)$, $((\exists x)\varphi) \in$ FOR;
- if $\varphi \in$ FOR and $t \in$ TERM then $([t]\varphi) \in$ FOR.

A formula of the form $p(t_1, \ldots, t_n)$ is called *atomic formula.*

We omit the parentheses if they are unnecessary: we use the already defined precedence but where the quantifiers have the highest.

The meaning of *free* and *bound* occurrence of variables are the usual ones. A *statement* is a formula in which all occurrences of all variables are bound. The *substitution* of a term t for a free variable x in the formula φ, denoted by $\varphi[t/x]$, is defined as usual: all free occurrences of x in φ are substituted by t with the proviso that free variables in t are not bound after the substitution. Observe that the term t replaces also the free variables x belonging to the terms of the modalities.[1]

V.2 Possible-worlds semantics

In a first-order Kripke interpretation each world is associated with a domain of quantification. We will not assume that domains are constant. The only restriction we put on them is that the domain of a world w is contained in the domain of all worlds reachable from w, i.e. domains are *increasing* (or *monotone*).[2] In each Kripke interpretation we will fix a non-empty set D of possible objects. The domain of each world will be a subset of D.

Definition V.2.1 (First-order Kripke interpretation) *Given a modal language $\mathcal{L}_{FO}$, a first-order Kripke interpretation M is an ordered tuple $\langle W, \mathcal{R}, D, \mathcal{J}, V\rangle$, where:*

- *W is a non-empty set of worlds;*
- *D is a non-empty set of objects;*
- *$\mathcal{J}$ is a function from W to non-empty subsets of D (it associates a domain with each world), satisfying the following condition: for all $w, w' \in W$, if $(w, w') \in \mathcal{R}$[3] then $\mathcal{J}(w) \subseteq \mathcal{J}(w')$;*

[1]For instance, $((\forall x)[t(y)]p(x, y))[a/y]$ is the formula $((\forall x)[t(a)]p(x, a))$.

[2]In particular, the Barcan formula $BF(t) : ((\forall x)[t]\varphi) \supset [t](\forall x)\varphi$ does not hold.

[3]That is, if there exists a parameter $d \in D$ such that $(w, w') \in \mathcal{R}_d$.

- *V is an* assignment function*, such that:*
  - *for each variable $x \in$ VAR of $\mathcal{L}_{FO}$, $V(x) \in D$;*
  - *for each function symbol $f \in \text{FUNC}^n$ of $\mathcal{L}_{FO}$, $V(f) \in D^n \rightarrow D$ and, for each world $w \in W$, the domain $\mathcal{J}(w)$ is closed with respect to the interpretation of f;*[4]
  - *for each predicate symbol $p \in \text{PRED}^n$ of $\mathcal{L}_{FO}$ and each world $w \in W$, $V(p, w) \subseteq D^n$, i.e., $V(p, w)$ is a set of n-tuples $\langle d_1, \ldots, d_n \rangle$, where each d_i is an element in D;*
- *$\mathcal{R}$ is the* accessibility relation*. It is* parameterized *with respect to domain elements, i.e. for each domain element $d \in D$ the accessibility relation $\mathcal{R}_d$ is a binary relation on W.*

Interpretation for terms in the domain is defined as usual from the interpretation of variables and function symbols. We say that M is based on *the frame $(W, \mathcal{R})$.*

We use $\mathcal{F}_{\mathcal{L}_{FO}}$ and $\mathcal{M}_{\mathcal{L}_{FO}}$ to denote the class of frame and the class of Kripke interpretations with $\mathcal{L}_{FO}$ as underlying language.

Let M be a Kripke interpretation, let $w \in W$ be a world, and let V be an assignment function. Then, we say that a formula φ of $\mathcal{L}_{FO}$ is satisfied by V in the Kripke interpretation M at w, denoted by $M, w \models^V \varphi$, if the following conditions hold:

- $M, w \models^V p(t_1, \ldots, t_n)$ iff $\langle V(t_1), \ldots, V(t_n) \rangle \in V(p, w)$;
- $M, w \models^V \neg\varphi$ iff $M, w \not\models^V \varphi$;
- $M, w \models^V \varphi \wedge \psi$ iff $M, w \models^V \varphi$ and $M, w \models^V \psi$;
- $M, w \models^V \varphi \vee \psi$ iff $M, w \models^V \varphi$ or $M, w \models^V \psi$;
- $M, w \models^V \varphi \supset \psi$ iff $M, w \not\models^V \varphi$ or $M, w \models^V \psi$;
- $M, w \models^V (\forall x)\varphi$ iff for every variable assignment V' that agrees with V everywhere except on x, and such that $V'(x) \in \mathcal{J}(w)$, $M, w \models^{V'} \varphi$;
- $M, w \models^V (\exists x)\varphi$ iff for some variable assignment V' that agrees with V everywhere except on x, and such that $V'(x) \in \mathcal{J}(w)$, $M, w \models^{V'} \varphi$;
- $M, w \models^V [t]\varphi$ iff for all $w' \in W$ such that $(w, w') \in \mathcal{R}_{V(t)}$, $M, w' \models^V \varphi$;
- $M, w \models^V \langle t \rangle\varphi$ iff there is a $w' \in W$ such that $(w, w') \in \mathcal{R}_{V(t)}$, $M, w' \models^V \varphi$.

[4]That is, for each n-ary function f and for $d_1, \ldots, d_n \in \mathcal{J}(w)$, $V(f)(d_1, \ldots, d_n) \in \mathcal{J}(w)$.

A formula φ of a language $\mathcal{L}_{FO}$ is *satisfiable in* a Kripke interpretation $M = \langle W, \mathcal{R}, D, \mathcal{J}, V\rangle$ if $M, w \models^V \varphi$ for some $w \in W$ with every term of φ interpreted in $\mathcal{J}(w)$. We say that φ is *valid in* M if $\neg\varphi$ is not satisfiable. Moreover, a formula φ is *satisfiable with respect to a class* $\mathcal{M}$ of Kripke interpretations if φ is satisfiable in some Kripke interpretation in $\mathcal{M}$, and it is *valid with respect to* $\mathcal{M}$ if it is valid in all Kripke interpretations in $\mathcal{M}$.

Remark V.2.1 Notice that, since the domain may change from a world to another, there is the problem of defining the satisfiability at a world w of a formula $\varphi(t)$ containing a term t whose interpretation is not in $\mathcal{J}(w)$. As mentioned by Fitting in [Fitting, 1983, pages 341-342], there are three intuitive choices to deal with this problem:

1. always take $\varphi(t)$ to be false in w;
2. leave the truth undetermined in w;
3. make no special restriction whatsoever.

Concerning choice 1), as Fitting mentions, Kripke has observed that imposing this requirement on atomic formulae leads to a modal logic in which the rule of substitution does not apply (see also [Hughes and Cresswell, 1996, pages 275-276]). Choice 2) has been made in [Hughes and Cresswell, 1968, Chapter 10]. In this case interpretations are three valued: the truth value of any formula in a world can be either true or false or undefined. Finally, choice 3) is the simplest one: first it does not put any special requirement on the valuation of formulae, provided that in defining validity and satisfiability of a formula, for each interpretation, only those worlds are considered such that the constants of the formulae have their interpretation in the domain of the world. Indeed, choice 2) and choice 3) are equivalent [Fitting, 1983; Hughes and Cresswell, 1996].

With regard to this we adopt choice 3 and we do not make any special restriction. However, when we define satisfiability and validity of a formula we look at the truth value of the formula in an interpretation at a certain world only if the interpretation of each term in the formula is in the domain of that world. Moreover, we require that functions map elements of a domain to elements of the same domain of that world.

Remark V.2.2 In general, when function symbols are present, each function symbol could be given a different interpretation at each different world. In the Kripke semantics above, however, function symbols are given the same interpretation in all possible worlds. As a consequence, closed terms have the same interpretation in all possible worlds (*rigid designators*). On the contrary, predicate symbols may have a different interpretation in each possible world. For a survey of the different systems for quantified modal logic see [Garson, 1984], while for more details on the characterization of first-order inclusion modal logics see [Gasquet, 1994].

As for the propositional case, we are interested in a particular subclass of Kripke interpretations. Given a predicative modal language $\mathcal{L}_{FO}$ and a set of inclusion axiom schemas $\mathcal{A}$, we are interested in first-order Kripke $\mathcal{A}$-interpretations, that is, first-order Kripke interpretations based on $\mathcal{A}$-inclusion frames as defined in Section II.3. We denote with $\mathcal{F}^{\mathcal{A}}_{\mathcal{L}_{FO}}$ the subset of $\mathcal{F}_{\mathcal{L}_{FO}}$ that consists of all $\mathcal{A}$-inclusion frames, with $\mathcal{M}^{\mathcal{A}}_{\mathcal{L}_{FO}}$ the subset of $\mathcal{M}_{\mathcal{L}_{FO}}$ of all Kripke $\mathcal{A}$-interpretations, and with $IP^{\mathcal{A}}_{\mathcal{L}_{FO}}$ the set of *inclusion properties* that a Kripke $\mathcal{A}$-interpretation must verify. We will use the already introduced notation of satisfiability and validity. Finally, we denote with $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$ the first-order inclusion modal logic determined by means of the set of axiom $\mathcal{A}$.

V.3 A predicate tableau calculus

In this section we extend the tableau calculus presented in Chapter III in order to deal with predicate case. However, for simplicity, in the following we will be concerned with a language containing:

- *only* constant symbols and *no* function symbols (we will call C its collection);
- the modalities are labeled as in the propositional case (with constant symbols) and *not* with terms.

Given a first-order modal language $\mathcal{L}_{FO}$, since the proofs in the tableau calculus have to deal with free variables, we extend the $\mathcal{L}_{FO}$ with *countably* many *new constant*, called *parameters* [Fitting, 1983, Chapter 7, Section 2]. These parameters are used, as in tableaux for classical predicate logic, as *witnesses* for existential quantifiers. We call the extended language $\overline{\mathcal{L}_{FO}}$. In particular, in order to deal with increasing domains, for each world constant symbol $w \in \mathcal{W}_C$, we extend $\mathcal{L}_{FO}$ with a countable list P_w of new individual constant symbols, disjoint from those of $\mathcal{L}_{FO}$, and such that for each pair of distinct prefixes w and w' we have that P_w and $P_{w'}$ do not overlap [Fitting, 1993, Section 2.4].[5] We say $a_w \in P_w$ a *w-parameter*. Note that a proof of a formula of $\mathcal{L}_{FO}$ can make use of formulae of $\overline{\mathcal{L}_{FO}}$.

Universal formulae		*Existential formulae*	
γ	$\gamma_0(c)$	δ	$\delta_0(c)$
$\mathbf{T}(\forall x)\varphi$	$\mathbf{T}\varphi[c/x]$	$\mathbf{F}(\forall x)\varphi$	$\mathbf{F}\varphi[c/x]$
$\mathbf{F}(\exists x)\varphi$	$\mathbf{F}\varphi[c/x]$	$\mathbf{T}(\exists x)\varphi$	$\mathbf{T}\varphi[c/x]$

Figure V.1: Uniform notation for quantified formulae.

Now, we can add the extension rules for predicate logic quantifiers to those of propositional modal logic. The meaning of *proof* ($\mathcal{T}^{\mathcal{A}}_{\mathcal{L}_{FO}}$-provability) is trivially updated. We make use of the *uniform notation* for the quantified signed formulae given in Figure V.1.

[5] This is necessary because we deal with modal tableau system with *explicit accessibility*. Other methods, such as the cut-free sequent calculus in [Wallen, 1990, Section 2.1] and in [Baldoni *et al.*, 1997a, Section 6.1] or the tableau method in [Fitting, 1983, Chapter 7], do not need this trick because at any stage of a proof *only* the formulae of the current world are present.

Definition V.3.1 (Extension rules) *Let $\overline{\mathcal{L}_{FO}}$ be a modal language and let $\mathcal{A}$ be a set of inclusion axiom schemas, the extension rules (tableau rules) for $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$ are given in Figure III.2 and Figure V.2.*

$$\frac{w : \gamma}{w : \gamma_0(c)}\ \gamma\text{-rule} \qquad\qquad \frac{w : \delta}{w : \delta_0(a_w)}\ \delta\text{-rule}$$

c is a w-available world constant symbol

Engenvariable condition: a_w is a *w-parameter* that does not occur on the branch.

Figure V.2: Tableau rules for quantified formulae.

A formula of type γ is true at world w if $\gamma_0(c)$ is true for all constant symbols of the domain of w. Therefore, if $w : \gamma$ occurs on an open branch S, we can add $w : \gamma_0(c)$ to the end of that branch for any constant c which belongs to the domain of w. Now, since the domains are increasing, if w is *reachable* from a world w', that is there is a path $\xi(w', w)$ in S, then, the constant c used in $w : \gamma_0(c)$ can be either a constant symbol of C or it is a w-parameter or w'-parameter in S. We say a such constant *w-available.*

The interpretation of the extension rule for formulae of type δ is the usual one. In order to express the meaning of a formula of type δ, there should be something making δ true, we use a parameter never used before on the branch to substitute the existential quantified variable.

Example V.3.1 *(Barcan formula)* In $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$, with $\mathcal{A}$ any set of inclusion axioms, the following instance of the Barcan formula $BF(t) : ((\forall x)[t]\varphi) \supset [t](\forall x)\varphi$ is not provable:

1. $i : \mathbf{F}(((\forall x)[t]p(x)) \supset [t](\forall x)p(x))$
2. $i : \mathbf{T}(\forall x)[t]p(x)$
3. $i : \mathbf{F}[t](\forall x)p(x)$
4. $w_1 : \mathbf{F}(\forall x)p(x)$
5. $i\ \rho_t\ w_1$
6. $w_1 : \mathbf{F}p(a_{w_1})$

Explanation: *1.*: an instance of the Barcan formula; *2. and 3.*: from 1., by α-rule; *4. and 5.*: from 3., by application of π-rule; *6.*: form 4., by application of δ-rule. Since the constant symbol a_{w_1} is not i-available the branch remains open.

Example V.3.2 *(Converse of Barcan formula)* In $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$, with $\mathcal{A}$ any set of inclusion axioms, the following instance of the converse of Barcan formula $BF_c(t) : [t](\forall x)\varphi \supset ((\forall x)[t]\varphi)$ is provable:

1. $i : \mathbf{F}([t](\forall x)p(x) \supset ((\forall x)[t]p(x)))$
2. $i : \mathbf{T}[t](\forall x)p(x)$
3. $i : \mathbf{F}(\forall x)[t]p(x)$

4. $i : \mathbf{F}[t]p(a_i)$
5. $w_1 : \mathbf{F}p(a_i)$
6. $i \, \rho_t \, w_1$
7. $w_1 : \mathbf{T}(\forall x)p(x)$
8. $w_1 : \mathbf{T}p(a_i)$

   $\times$

Explanation: *1.*: an instance of the converse Barcan formula; *2. and 3.*: from 1., by α-rule; *4.*: from 3., by application of δ-rule; *5. and 6.*: form 4., by application of π-rule; *7.*: from 2. and 6., by application of ν-rule; *8.*: from 7., by application of γ-rule, branch closes.

Theorem V.3.1 (Soundness and Completeness) *Let $\mathcal{L}_{FO}$ be a predicative modal language and let $\mathcal{A}$ be a set of inclusion axiom schemas, a formula φ of $\mathcal{L}_{FO}$ is $\mathcal{A}$-valid if and only if φ is $\mathcal{T}^{\mathcal{A}}_{\mathcal{L}_{FO}}$-provable.*

Proof. Both the proofs of the soundness and completeness are based on the same technique used for the ones for propositional case given in Chapter III. In particular, we can note that:

- An $\mathcal{A}$-mapping I (see page 28) must map both prefixes and constant symbols of the language to the worlds and constants of some first-order Kripke $\mathcal{A}$-interpretation.
- In the systematic tableau procedure, in order to deal with the prefixed signed formulae of form $w : \gamma$ and to make sure $w : \gamma_0(c)$ has been introduced for each constant symbol c that occurs on the considered branch, we use the same trick adopted for formulae of type ν^t. Then, whenever we apply γ-rule to a formula of type γ, we add a *fresh* occurrence of it at the end of that branch.
- In the line of [Fitting, 1983], we can update the Definition III.3.2 of set of prefixed signed and accessibility relation formulae $\mathcal{A}$-downward satured as follows:

  7. if $w : \gamma \in S$, then $w : \gamma_0(c) \in S$ for all $c \in C$ and all $c \in \cup_{w' \in S} P_{w'}$ such that there exists a path $\xi(w, w')$ in S;
  8. if $w : \delta \in S$, then $w : \delta_0(c) \in S$ for some w-parameter $c \in P_w$.

- From an open $\mathcal{A}$-downward-satured branch S we define a first-order canonical model $\mathcal{M}^{\mathcal{A}}_c$ as follows. The set of worlds and the set of accessibility relations are defined as we did in the propositional case. D is C added to $\cup_{w \in S} P_w$, the domain on S is $C \cup P_w$ together $\cup_{w' \in S} P_{w'}$ such that w is reachable by w'. Each constant symbol and parameter is interpreted as naming itself. Finally, for each predicative symbol $p \in \text{PRED}^n$ and world w used in S, we define $V(p, w) = \{p(c_1, \ldots, c_n) \mid w : p(c_1, \ldots, c_n) \in S\}$.

$\square$

Chapter VI

Towards a wider class of logics

In this chapter, we extend the tableau calculus of Chapter III in order to deal with the class of normal multimodal logics proposed in [Catach, 1988]. This class is determined by the *interaction axiom* $G^{a,b,c,d}$, called a, b, c, d-incestuality axiom. It includes most of the modal and multimodal systems studied in the literature. Moreover, modal operator can be labeled by *complex* parameters, i.e. built from atomic ones, using an operator of *composition* and an operator of *union.*

VI.1 Syntax and possible-worlds semantics

Syntax

Let us extend the alphabet of the language for propositional multimodal logics of Section II.1 adding the following symbols:

- a binary operator "$\cup$" (*union*);
- a binary operator ";" (*composition*);
- the symbol "ε" (the *neutral element* w.r.t. the composition).

The operators "$\cup$" and ";" allow to built up new labels for modal operators starting from the atomic ones in MOD. More formally, we define the set LABELS as the least set that satisfies the following conditions:

- $\varepsilon \in$ LABELS;
- MOD $\subseteq$ LABELS;
- if $t, t' \in$ LABELS then $(t; t')$ and $(t \cup t')$ are in LABELS.[1]

[1]For readability, we omit parentheses if they are unnecessary: we give "$\cup$" lower precedence than ";".

The set FOR of formulae of a modal propositional language $\mathcal{L}$ is defined to be the least set that satisfies the following conditions:

- VAR $\subseteq$ FOR;
- if $\varphi, \psi \in$ FOR then $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \supset \psi) \in$ FOR;
- if $\varphi \in$ FOR and $t \in$ LABELS then $([t]\varphi) \in$ FOR.

Therefore, we allow modal operators labeled with expressions built by the operators union and composition on the atomic labels MOD together the empty label ε. As usual, $\langle t\rangle\varphi$ stands for $\neg[t]\neg\varphi$. Examples of modalized formulae are $[t; t' \cup t'' \cup \varepsilon]\varphi$ and $\langle t; t'\rangle\varphi$. Indeed, the empty label, union, and composition can be though as a shorthand, as stated by the following definitions:

- $[\varepsilon]\varphi =_{\text{Def.}} \varphi$;
- $[t \cup t']\varphi =_{\text{Def.}} [t]\varphi \wedge [t']\varphi$;
- $[t; t']\varphi =_{\text{Def.}} [t][t']\varphi$.

For instance, the above modalized formulae are equivalent to $[t][t']\varphi \wedge [t'']\varphi \wedge \varphi$ and $\langle t\rangle\langle t'\rangle\varphi$, respectively.

Possible-worlds semantics

In order to define the meaning of a formula, we have introduce in the previous chapter the notion of *Kripke interpretation*. Formally, a Kripke interpretation M is a triple $\langle W, \mathcal{R}, V\rangle$, consisting of a non-empty set W of "*possible worlds*", a mapping $\mathcal{R}$ from MOD to the powerset of $W \times W$ (it assigns to each atomic label of MOD some binary relation on W), and a *valuation function* V, that is a mapping from $W \times$ VAR to the set $\{\mathbf{T}, \mathbf{F}\}$. Here, in order to deal with any label $t \in$ LABELS, we extend the mapping $\mathcal{R}$ inductively as follows:

- $\mathcal{R}_\varepsilon = I$, where $I = \{(w, w) \mid w \in W\}$ (the *identity* relation);
- $\mathcal{R}_{t;t'} = \mathcal{R}_t \circ \mathcal{R}_{t'}$, where "$\circ$" denotes the composition of binary relations;
- $\mathcal{R}_{t \cup t'} = \mathcal{R}_t \cup \mathcal{R}_{t'}$, where "$\cup$" denotes the union of binary relations.

We say that $\mathcal{R}_t$ is the *accessibility relation* of the modality $[t]$ and w' is *accessible* from w *by means of* $\mathcal{R}_t$ if $(w, w') \in \mathcal{R}_t$.

The meaning of a formula is given by means of a *satisfiability relation*, denoted by $\models$, as already seen.

VI.2 Incestual modal logics

In [Catach, 1988] *incestual modal logics* the class of normal modal logics obtained by taking axiom systems containing:

$$[\varepsilon]\varphi \quad \Leftrightarrow \quad \varphi \tag{VI.1}$$
$$[t;t']\varphi \quad \Leftrightarrow \quad [t][t']\varphi \tag{VI.2}$$
$$[t \cup t']\varphi \quad \Leftrightarrow \quad [t]\varphi \wedge [t']\varphi \tag{VI.3}$$

where $t, t' \in$ LABELS, and a finite set of a, b, c, d*-incestual axiom schemas*, that is axiom schemas of the form:

$$G^{a,b,c,d} : \langle a \rangle [b]\varphi \supset [c]\langle d \rangle \varphi$$

where a, b, c, and d belong to LABELS. Given a modal language $\mathcal{L}$ and a set $\mathcal{G}$ of incestual axiom schemas, we denote with $\mathcal{S}_{\mathcal{L}}^{\mathcal{G}}$ the axiom system $\mathcal{S}_{\mathcal{L}}$[2] extended with $\mathcal{G}$ together the axioms (VI.1), (VI.2), and (VI.3). We use $\mathcal{I}_{\mathcal{L}}^{\mathcal{G}}$ to denote the incestual modal logics determined by $\mathcal{S}_{\mathcal{L}}^{\mathcal{G}}$. As we will see, the incestual axioms also determine inclusion properties on the accessibility relations.

As it is remarked in [Catach, 1988], the fact that a, b, c, and d of an incestual axiom schema may be arbitrary expressions built from atomic labels using the composition and union operators, makes axiom $G^{a,b,c,d}$ very general. In particular, it covers the axiom [Chellas, 1980; Hughes and Cresswell, 1984]:

$$G^{k,l,m,n} : \Diamond^k \Box^l \varphi \supset \Box^m \Diamond^n \varphi$$

where $k, l, m, n \geq 0$, and, therefore, it covers the traditional axiom schemas. Furthermore, it captures many axiom schemas which can express *interaction* between different modal operators (see Figure VI.1). Note that, the inclusion axiom schema $[t_1] \dots [t_n]\varphi \supset [s_1] \dots [s_m]\varphi$ is an instance of the a, b, c, d-incestual axiom schema too. In fact, it is enough to take $a = \varepsilon$, $b = t_1; \dots ; t_n$, $c = s_1; \dots ; s_m$, and $d = \varepsilon$.

All the fifteen modal systems obtained combining the axioms T, D, B, 4, and 5 [Chellas, 1980; Hughes and Cresswell, 1996] and their multimodal versions [Halpern and Moses, 1992] are incestual modal logics, as well as the extensions of K_n and $S4_n$ with interaction axioms of with agent "any fool" [Enjalbert and Fariñas del Cerro, 1989].

Example VI.2.1 *(The wise men puzzle)* The problem is again the well-known *three wise men puzzle* already presented in Example II.3.4. We call back briefly the formulation. Note that, in order to avoid introducing many variants of the same formulae and axioms for the different wise men, as a shorthand, we use the metavariables X, Y, and Z, where $X, Y, Z \in \{a, b, c\}$ and $X \neq Y$, $Y \neq Z$, and $X \neq Z$:

(1) $[fool](ws(a) \vee ws(b) \vee ws(c))$
(2) $[fool](ws(X) \supset [Y]ws(X))$
(3) $[fool](\neg ws(X) \supset [Y]\neg ws(X))$

[2]See Chapter II.

axiom name		*axiom schema*	*incestual schema*
reflexivity	$T(t)$	$[t]\varphi \supset \varphi$	$G^{\varepsilon,t,\varepsilon,\varepsilon}$
seriality	$D(t)$	$[t]\varphi \supset \langle t\rangle\varphi$	$G^{\varepsilon,t,\varepsilon,t}$
symmetry	$B(t)$	$\langle t\rangle[t]\varphi \supset \varphi$	$G^{t,t,\varepsilon,\varepsilon}$
transitivity	$4(t)$	$[t]\varphi \supset [t][t]\varphi$	$G^{\varepsilon,t,(t;t),\varepsilon}$
euclideanity	$5(t)$	$\langle t\rangle\varphi \supset [t]\langle t\rangle\varphi$	$G^{t,\varepsilon,t,t}$
determinism	$\delta(t)$	$\langle t\rangle\varphi \supset [t]\varphi$	$G^{t,\varepsilon,t,\varepsilon}$
inclusion	$I(t,t')$	$[t]\varphi \supset [t']\varphi$	$G^{\varepsilon,t,t',\varepsilon}$
mutual transitivity	$4M(t,t')$	$[t]\varphi \supset [t'][t]\varphi$	$G^{\varepsilon,t,(t';t),\varepsilon}$
persistence	$P(t,t')$	$[t][t']\varphi \supset [t'][t]\varphi$	$G^{\varepsilon,(t;t'),(t';t),\varepsilon}$
relative inclusion	$I_r(t,t',t'')$	$[t]\varphi \supset ([t']\varphi \supset [t'']\varphi)$	$G^{\varepsilon,(t\cup t'),t'',\varepsilon}$
semi-adjunction	$B(t,t')$	$\varphi \supset [t]\langle t'\rangle\varphi$	$G^{\varepsilon,\varepsilon,t,t'}$
mutual seriality	$D(t,t')$	$[t]\varphi \supset \langle t'\rangle\varphi$	$G^{\varepsilon,t,\varepsilon,t'}$
union		$[t]\varphi \supset [t']\varphi \wedge [t'']\varphi$	$G^{\varepsilon,t,(t'\cup t''),\varepsilon}$
		$[t']\varphi \wedge [t'']\varphi \supset [t]\varphi$	$G^{\varepsilon,(t'\cup t''),t,\varepsilon}$
composition		$[t]\varphi \supset [t'][t'']\varphi$	$G^{\varepsilon,t,(t';t''),\varepsilon}$
		$[t'][t'']\varphi \supset [t]\varphi$	$G^{\varepsilon,(t';t''),t,\varepsilon}$

Figure VI.1: Some well-known axiom schemas included by the incestual axioms.

where $ws(X)$ means that the wise man X has a white spot on his forehead and $[X]$ is a modal operator of type K. The formulae above are all preceded by the modal operator $[fool]$ of type $S4$ which captures to the information common to all wise men. That is, it is axiomatized by the axioms:

(A_1) $T(fool) : [fool]\varphi \supset \varphi$
(A_2) $4(fool) : [fool]\varphi \supset [fool][fool]\varphi$
(A_3) $I(fool, a) : [fool]\varphi \supset [a]\varphi$
(A_4) $I(fool, b) : [fool]\varphi \supset [b]\varphi$
(A_5) $I(fool, c) : [fool]\varphi \supset [c]\varphi$

The formulae (1) says that at least one of the wise men has a white spot, whereas formulae (2) and (3) means that whenever one of them has (not) a white spot, the others know this fact. Moreover, whenever a wise man does (not) know something the others know that he does not know this. That is, the following axiom is assumed:

(A_6) $\neg[X]\varphi \supset [Y]\neg[X]\varphi$ (i.e. $\langle X\rangle\varphi[\varepsilon] \supset [Y]\langle X\rangle\varphi$)
(A_7) $[X]\varphi \supset [Y][X]\varphi$ (i.e. $\langle\varepsilon\rangle[X]\varphi \supset [Y;X]\langle\varepsilon\rangle\varphi$)

From this formalization and the fact that neither a nor b know if they have a white spot on their forehead:

(4) $\neg[a]ws(a)$
(5) $\neg[b]ws(b)$

follows that c knows that he has a white spot:

(6) $[c]ws(b)$

Note that, differently than the formulation of Example II.3.4, here we do not need to express directly the information that if someone does not know if his spot is white then the others knows that he does not know it (formulae (3) and (4) of Example II.3.4) but they are inferred by the axiom (A_6).

Definition VI.2.1 (Incestual frame) *Let $\mathcal{L}$ be a propositional modal language and let $\mathcal{G}$ be a set of incestual axiom schemas. Then, a frame $F \in \mathcal{F}_{\mathcal{L}}$ is a $\mathcal{G}$-*incestual frame *if and only if for each axiom $G^{a,b,c,d} \in \mathcal{G}$ the following inclusion property (called in [Catach, 1988] a,b,c,d-*incestual property*) on the accessibility relations holds:*

$$\mathcal{R}_b \circ \mathcal{R}_d^{-1} \supseteq \mathcal{R}_a^{-1} \circ \mathcal{R}_c \tag{VI.4}$$

where R is the mapping defined at page 54 and $\mathcal{R}_t^{-1}$ is the inverse relation of $\mathcal{R}_t$. We call $IP_{\mathcal{L}}^{\mathcal{G}}$ the set of incestual properties determined by $\mathcal{G}$.

In other worlds: "*if $(w, w') \in \mathcal{R}_a$ and $(w, w'') \in \mathcal{R}_c$ then there exists w^* such that $(w', w^*) \in \mathcal{R}_b$ and $(w'', w^*) \in \mathcal{R}_d$*":

$$\frac{\forall w, w', w'' \in W \ (w, w') \in \mathcal{R}_a \land (w, w'') \in \mathcal{R}_c}{\exists w^* \in W \ (w', w^*) \in \mathcal{R}_b \land (w'', w^*) \in \mathcal{R}_d} \tag{VI.5}$$

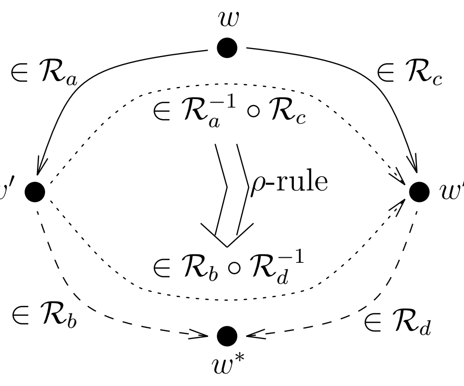

Figure VI.2: a,b,c,d-incestual property. This property is named *incestual* because the offspring of a common parent have themselves an offspring in common [Chellas, 1980].

Figure VI.2 shows pictorially the a,b,c,d-incestual property.

We denote with $\mathcal{F}_{\mathcal{L}}^{\mathcal{G}}$ the set of $\mathcal{G}$-frame and with $\mathcal{M}_{\mathcal{L}}^{\mathcal{G}}$ the set of Kripke interpretations based on a $\mathcal{G}$-frame (Kripke $\mathcal{G}$-interpretations). The definitions of *satisfiability relation* "$\models_{\mathcal{G}}$ ", *$\mathcal{G}$-satisfiability*, *$\mathcal{G}$-validity* are the usual ones.

Catach proved that a multimodal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{G}}$ is *determined* by the class of Kripke $\mathcal{G}$-interpretations (the completeness proof uses the standard canonical model construction).

Theorem VI.2.1 ([Catach, 1988]) *Let $\mathcal{L}$ be a propositional modal language and let $\mathcal{G}$ be a finite set of incestual axiom schemas. Then, $\mathcal{S}_{\mathcal{L}}^{\mathcal{G}}$ is a sound and complete axiomatization with respect to $\mathcal{M}_{\mathcal{L}}^{\mathcal{G}}$.*

Remark VI.2.1 Despite the fact that the class of incestual modal logics includes a wide class of multimodal systems, it is worth noting that *no* set of inclusion properties of the form (VI.4) can characterize the the *induction axiom* that define both the *iteration* operator "$*$" of dynamic logic [Harel, 1984] and the *common knowledge* operator "$\mathcal{C}$" [Genesereth and Nilsson, 1987; Halpern and Moses, 1992]. In fact, let us consider the axioms:

$$[b]\varphi \supset \varphi \wedge [a][b]\varphi \tag{VI.6}$$

$$\varphi \wedge [b](\varphi \supset [a]\varphi) \supset [b]\varphi \tag{VI.7}$$

then, it is easy to see that the modal operator $[b]$ represents both $[a^*]$ of dynamic logic and the common knowledge operator (when a is the only agent). Axiom (VI.6) is an incestual axiom (it is equal to $\langle\varepsilon\rangle[b]\varphi \supset [\varepsilon \cup a; b]\langle\varepsilon\rangle\varphi$) but axiom (VI.7), the *induction axiom*, is not (see also [Catach, 1988]). From a semantics point of view, the axioms (VI.6) and (VI.7) are characterized by the class of Kripke interpretations in which the relation $\mathcal{R}_b$ is equal to $\mathcal{R}_a^*$ [Kozen and Tiuryn, 1990; Halpern and Moses, 1992] (i.e. the *reflexive* and *transitive* closure of $\mathcal{R}_a$). On the contrary, incestual axioms are not strong enough to capture $\mathcal{R}_a^*$.

Indeed, axiom (VI.6) can be characterized by the inclusion properties $\mathcal{R}_b \supseteq I \cup \mathcal{R}_a \circ \mathcal{R}_b$, from which, by some easy transformations, we have $\mathcal{R}_b \supseteq \mathcal{R}_a^*$. Unfortunately, the converse of axiom (VI.6), that is

$$\varphi \wedge [a][b]\varphi \supset [b]\varphi \tag{VI.8}$$

does not capture the converse inclusion relation $\mathcal{R}_a^* \supseteq \mathcal{R}_b$ [Catach, 1988]. The modal systems which contain the axioms (VI.6) and (VI.8) are sound and complete with respect to the class of Kripke interpretations for which the relation

$$\mathcal{R}_b = I \cup \mathcal{R}_a \circ \mathcal{R}_b$$

holds but this does not mean that $\mathcal{R}_b$ is equal to $\mathcal{R}_a^*$. In fact, let us define the function

$$F(X) = I \cup \mathcal{R}_a \circ X$$

then, $\mathcal{R}_b$ is equal to a fixpoint of F. Now, F is monotone and continuous and, then, the least fixed point of F exists and it is equal to $\cup_{k\in\omega}F^k(\emptyset)$, that corresponds to $\mathcal{R}_a^*$. However, in general, this is not the only fixpoint of F.[3]

[3]Let us consider, for instance, $W = \{w_1, w_2\}$ and, then, $I = \{(w_1, w_1), (w_2, w_2)\}$. Assume that $\mathcal{R}_a = I$, the least fixpoint of F is $\mathcal{R}_a^*$, that is I itself. Now, consider the set $B = \{(w_1, w_1), (w_2, w_2), (w_1, w_2)\}$ then, $F(B) = I \cup \mathcal{R}_a \circ B$ and, since we have assumed $\mathcal{R}_a = I$, $F(B) = I \cup I \circ B$. Since $I \circ B = B$ and $I \cup B = B$, we have that $F(B) = B$, that is B is a fixpoint of F but $B \neq I$ (indeed, $\mathcal{R}_a^* = \mathcal{R}_a \not\supseteq B$).

VI.3 A tableau calculus

The tableau calculus for incestual modal logics extends the one presented in Chapter III. A tableau is a labeled tree where each node consists of a *prefixed signed formula* or of an *accessibility relation formula.* Intuitively, each tableau branch corresponds to the construction of a Kripke interpretation that satisfies the formulae that belong to it. However, in order to deal with arbitrary expressions as labels of modal operators, we need to extend the notion of accessibility relation formula.

Definition VI.3.1 *Let $\mathcal{L}$ be a propositional modal language, an* accessibility relation formula $w\ \rho_t\ w'$*, where* $t \in$ LABELS,[4] *is a binary relation between prefixes of* $\mathcal{W}_C$*.*

We say that an accessibility relation formula $w\ \rho_t\ w'$ is *true* in a tableau branch if it belongs to that branch. Moreover, the relation ρ_ε on a branch defines an equivalence relation among prefixes: when $w\ \rho_\varepsilon\ w'$ holds, w and w' can be regarded as representing the same worlds. By taking the reflexive, transitive and symmetric closure of the relation ρ_ε we define an equivalence relation among worlds. We denote by $\overline{w}$ the equivalence class of w with respect to this equivalence relation. A formula $w : \mathbf{T}\varphi$ ($w : \mathbf{F}\varphi$) on a branch of a tableau means that the formula φ is *true* (*false*) at the world $\overline{w}$ in the Kripke interpretation associated with that branch.

We say that a prefix w is *used* on a tableau branch if it occurs on the branch in some accessibility relation formula, otherwise we say that the prefix w is *new.* Moreover, given a label t, we say that an accessibility relation formula $w\ \rho_t\ w'$ is *available on* a branch S of a tableau if one of the following conditions hold:

1. $t = \varepsilon$ and $w = w'$;

2. $w_1 \in \overline{w}$, $w_2 \in \overline{w'}$ and $w_1\ \rho_t\ w_2$ is true in S;

3. $t = t'; t''$ and both $w\ \rho_{t'}\ w''$ and $w''\ \rho_{t''}\ w'$ are available on S, for some w'' used on the branch S;

4. $t = t' \cup t''$ and either $w\ \rho_{t'}\ w'$ is available on S or $w\ \rho_{t''}\ w'$ is available on S.

Note that, if an accessibility relation formula is true in a tableau branch, it is also available on it (as a special case of the condition 2 above). Moreover, for any world w on a given branch, $w\ \rho_\varepsilon\ w$ is always available (condition 1). Intuitively, $w\ \rho_t\ w'$ available on a branch of a tableau means that, in the Kripke interpretation associated with that branch, $(\overline{w}, \overline{w'}) \in \mathcal{R}_t$.

Definition VI.3.2 (Extension rules) *Let $\mathcal{L}$ be a propositional modal language and let $\mathcal{G}$ be a set of incestual axioms, the extension rules (tableau rules) for $\mathcal{I}^{\mathcal{G}}_{\mathcal{L}}$ are given in Figure VI.3.*

$$\frac{w : \alpha}{\begin{array}{c} w : \alpha_1 \\ w : \alpha_2 \end{array}}\ \alpha\text{-rule} \qquad\qquad \frac{w : \beta}{w : \beta_1 \mid w : \beta_2}\ \beta\text{-rule}$$

$$\frac{w : \nu^t \quad w\ \rho_t\ w'}{w' : \nu^t_0}\ \nu\text{-rule} \qquad\qquad \frac{w : \pi^t}{\begin{array}{c} w' : \pi^t_0 \\ w\ \rho_t\ w' \end{array}}\ \pi\text{-rule}$$

where $w\ \rho_t\ w'$ is *available* on the branch where w' is *new* on the branch

$$\frac{w\ \rho_{t;t'}\ w'}{\begin{array}{c} w\ \rho_t\ w'' \\ w''\ \rho_{t'}\ w' \end{array}}\ \rho_\alpha\text{-rule} \qquad\qquad \frac{w\ \rho_{t \cup t'}\ w'}{w\ \rho_t\ w' \mid w\ \rho_{t'}\ w'}\ \rho_\beta\text{-rule}$$

where w'' is *new* on the branch

$$\frac{w\ \rho_a\ w' \quad w\ \rho_c\ w''}{\begin{array}{c} w'\ \rho_b\ w^* \\ w''\ \rho_d\ w^* \end{array}}\ \rho\text{-rule}$$

where $w\ \rho_a\ w'$ and $w\ \rho_c\ w''$ are *available* on the branch,
w^* is *new* on the branch, and $\langle a \rangle [b]\varphi \supset [c]\langle d \rangle \varphi \in \mathcal{G}$

Figure VI.3: Tableau rules for propositional incestual modal logics.

The interpretation of the α, β, ν, and π rule is the same already seen in the previous chapters, the only remark is that, now, the label t of a formula ν^t or π^t can be an arbitrarily complex expression.

Case ρ_α-rule. If an accessibility relation formula $w\ \rho_{t;t'}\ w'$ is true in a tableau branch then, $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t;t'}$ holds in the Kripke interpretation represented by that branch. Therefore, $(\overline{w}, \overline{w'}) \in \mathcal{R}_t \circ \mathcal{R}_{t'}$ and, hence, there exists a world w'' such that $(\overline{w}, \overline{w''}) \in \mathcal{R}_t$ and $(\overline{w''}, \overline{w'}) \in \mathcal{R}_{t'}$. That is, $w\ \rho_t\ w''$ and $w''\ \rho_{t'}\ w'$ are true in that branch.

Case ρ_β-rule. If an accessibility relation formula $w\ \rho_{t \cup t'}\ w'$ is true in a tableau branch then, $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t \cup t'}$ holds in the Kripke interpretation represented by that branch. Therefore, $(\overline{w}, \overline{w'}) \in \mathcal{R}_t \cup \mathcal{R}_{t'}$ and, hence, $(\overline{w}, \overline{w'}) \in \mathcal{R}_t$ or $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t'}$. That is, $w\ \rho_t\ w'$ or $w\ \rho_{t'}\ w'$ is true in that branch.

Finally, the intuitive meaning of the ρ-rule is similar to the one of the calculus for inclusion modal logics and it allows us to deal with any incestual axiom in an uniform way. Let us suppose, for instance, that $\langle a \rangle [b]\varphi \supset [c]\langle d \rangle \varphi \in \mathcal{G}$ in our modal logic $\mathcal{I}^{\mathcal{G}}_{\mathcal{L}}$. If $w\ \rho_a\ w'$ and $w\ \rho_c\ w''$ are available on a tableau branch then, $(w, w') \in \mathcal{R}_a$ and $(w, w'') \in \mathcal{R}_c$ in the Kripke interpretation associated to that branch. Since the incestual axiom $\langle a \rangle [b]\varphi \supset [c]\langle d \rangle \varphi \in \mathcal{G}$ then, the corresponding a, b, c, d-incestual property (VI.5) must hold, that is, there exists a world w^* such that $(w', w^*) \in \mathcal{R}_b$ and $(w'', w^*) \in \mathcal{R}_d$. Hence, $w'\ \rho_b\ w^*$ and $w''\ \rho_d\ w^*$ are true in that Kripke interpretation for some new prefix w^* (see

[4]Instead of MOD!

Figure VI.2). Again the ρ-rule can be regarded as a *rewriting* rule which creates new paths among worlds according to the inclusion properties of the incestual modal logic.

We say that a tableau branch is *closed* if it contains $w : \mathbf{T}\varphi$ and $w' : \mathbf{F}\varphi$ for some formula φ such that $\overline{w} = \overline{w'}$. A tableau is *closed* if *every* branch in it is closed.

Definition VI.3.3 *Let $\mathcal{L}$ be a modal language and let $\mathcal{G}$ a set of incestual axioms. Then, given a formula φ of $\mathcal{L}$, we say that a closed tableau for $i : \mathbf{F}\varphi$, using the tableau rules of Figure VI.3, is a* proof *of φ (φ is $\mathcal{T}^{\mathcal{G}}_{\mathcal{L}}$-*provable*).*

Let us see some examples of derivations.

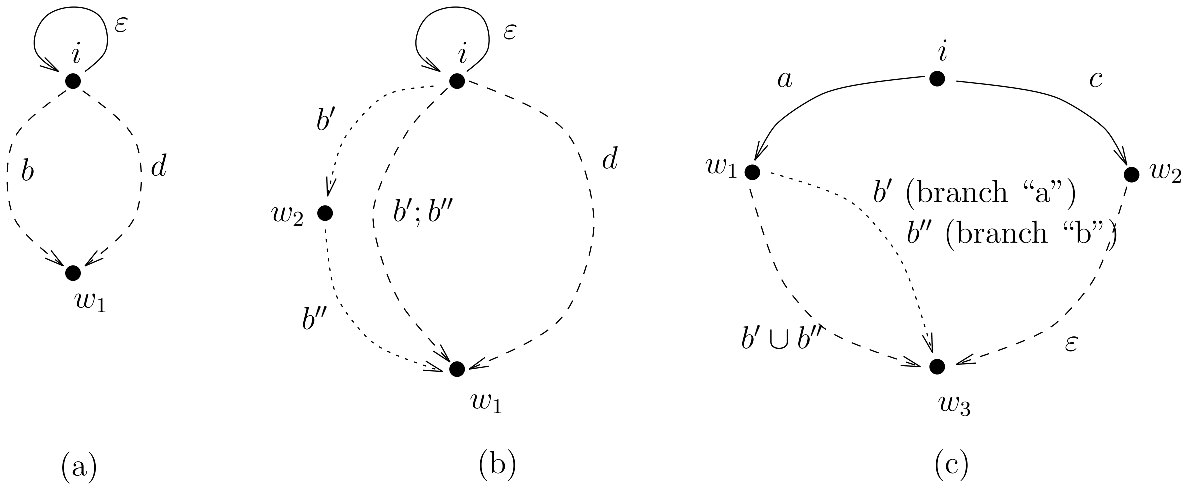

Figure VI.4: Kripke $\mathcal{G}$-interpretation constructions of Example VI.3.1, VI.3.2, and VI.3.3.

Example VI.3.1 Let us consider the incestual modal logic $\mathcal{I}^{\mathcal{G}}_{\mathcal{L}}$ where $\mathcal{G}$ that consists of the axiom schema $\langle\varepsilon\rangle[b] \supset \langle\varepsilon\rangle[d]$. Then, the formula $[b]p \supset \langle d\rangle p$ has a tableau proof (see also Figure VI.4(a)):

1.	$i : \mathbf{F}([b]p \supset \langle d\rangle p)$
2.	$i : \mathbf{T}[b]p$
3.	$i : \mathbf{F}\langle d\rangle p$
4.	$i\ \rho_b\ w_1$
5.	$i\ \rho_d\ w_1$
6.	$w_1 : \mathbf{T}p$
7.	$w_1 : \mathbf{F}p$
	$\times$

Explanation: *1.*: the goal; *2. and 3.*: from 1., by α-rule; *4. and 5.*: since $i\ \rho_\varepsilon\ i$ is available from axiom $G^{\varepsilon,b,\varepsilon,d}$, by ρ-rule; *6.*: from 2. and 4., by ν-rule; *7.*: from 3. and 5., by ν-rule, the branch closes due to steps 6. and 7.

Example VI.3.2 Let us consider the incestual modal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{G}}$ where $\mathcal{G}$ that consists of the axiom schema $\langle\varepsilon\rangle[b';b'']\varphi \supset [\varepsilon]\langle d\rangle\varphi$. Then, the formula $[b'][b'']p \supset \langle d\rangle p$ has a tableau proof (see also Figure VI.4(b)):

1.	$i : \mathbf{F}([b'][b'']p \supset \langle d\rangle p)$
2.	$i : \mathbf{T}[b'][b'']p$
3.	$i : \mathbf{F}\langle d\rangle p$
4.	$i\ \rho_d\ w_1$
5.	$i\ \rho_{b';b''}\ w_1$
6.	$i\ \rho_{b'}\ w_2$
7.	$w_2\ \rho_{b''}\ w_1$
8.	$w_2 : \mathbf{T}[b'']p$
9.	$w_1 : \mathbf{T}p$
10.	$w_1 : \mathbf{F}p$
	$\times$

Explanation: *1.*: the goal; *2. and 3.*: from 1., by α-rule; *4. and 5.*: by ρ-rule from axiom $\langle\varepsilon\rangle[b';b'']\varphi \supset [\varepsilon]\langle d\rangle\varphi$ since $i\ \rho_\varepsilon\ i$ is available; *6. and 7.*: from 5., by ρ_α-rule; *8.*: from 2. and 6., by ν-rule; *9.*: from 8. and 7., by ν-rule; *10.*: from 3. and 4., by ν-rule. The branch close due to steps 9. and 10.

Example VI.3.3 Let us consider the incestual modal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{G}}$ where $\mathcal{G}$ that consists of the axiom schema $\langle a\rangle[b' \cup b'']\varphi \supset [c]\langle\varepsilon\rangle\varphi$. Then, the formula $\langle a\rangle([b']p \wedge [b'']p) \supset [c]p$ has a tableau proof (see also Figure VI.4(c)) We denote with "a" and "b" the two branches which are created by the application of ρ_β-rule to step 10.

1.	$i : \mathbf{F}\langle a\rangle([b']p \wedge [b'']p) \supset [c]p$	
2.	$i : \mathbf{T}\langle a\rangle([b']p \wedge [b'']p)$	
3.	$i : \mathbf{F}[c]p$	
4.	$w_1 : \mathbf{T}([b']p \wedge [b'']p)$	
5.	$i\ \rho_a\ w_1$	
6.	$w_1 : \mathbf{T}[b']p$	
7.	$w_1 : \mathbf{T}[b'']p$	
8.	$w_2 : \mathbf{F}p$	
9.	$i\ \rho_c\ w_2$	
10.	$w_1\ \rho_{b' \cup b''}\ w_3$	
11.	$w_2\ \rho_\varepsilon\ w_3$	
12a.		$w_1\ \rho_{b'}\ w_3$
13a.		$w_3 : \mathbf{T}p$
		$\times$
12b.		$w_1\ \rho_{b''}\ w_3$
13b.		$w_3 : \mathbf{T}p$
		$\times$

Explanation: *1.*: the goal; *2. and 3.*: from 1., by α-rule; *4. and 5.*: from 2., by π-rule; *6. and 7.*: from 4., by α-rule; *8. and 9.*: from 3., by π-rule; *10. and 11.*: by ρ-rule from axiom $\langle a\rangle[b' \cup b'']\varphi \supset [c]\langle\varepsilon\rangle\varphi$ since $i\ \rho_a\ w_1$ and $i\ \rho_c\ w_2$ are available; *12a. and 12b.*: from 10., by ρ_β-rule;

13a.: from 6. and 12a., by ν-rule, *13b.*: from 7. and 12b., by ν-rule. Since $\overline{w_2} = \overline{w_3}$ ($w_2\ \rho_\varepsilon w_3$ belongs to the branch at step 11.) the branchs "a" and "b" close due to step 8. and steps 13a. and 13b., respectively.

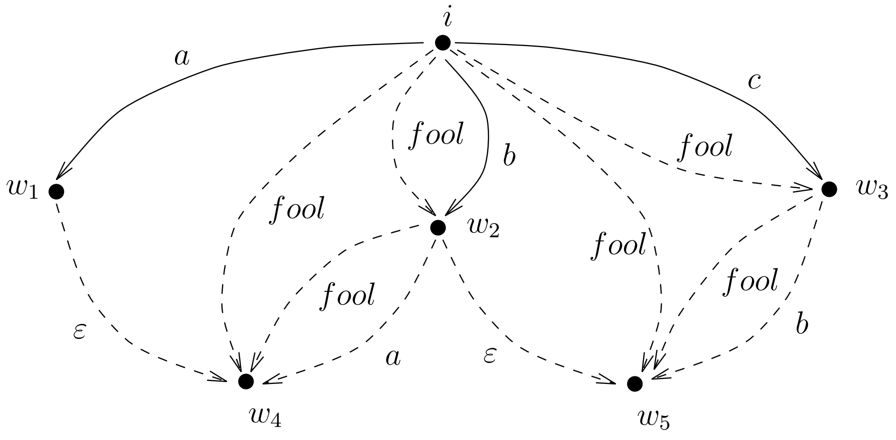

Figure VI.5: Kripke $\mathcal{G}$-interpretation construction of Example VI.3.4.

Example VI.3.4 *(The wise men puzzle)* We prove the formula (6) in Example VI.2.1 from the set of formulae (1)-(5). Figure VI.5 shows pictorially the counter-model construction.

1. $i : \mathbf{T}[fool](ws(a) \vee ws(b) \vee ws(c))$
2. $i : \mathbf{T}[fool](\neg ws(b) \supset [a]\neg ws(b))$
3. $i : \mathbf{T}[fool](\neg ws(c) \supset [a]\neg ws(c))$
4. $i : \mathbf{T}[fool](\neg ws(c) \supset [b]\neg ws(c))$
5. $i : \mathbf{T}\neg[a]ws(a)$
6. $i : \mathbf{T}\neg[b]ws(b)$
7. $i : \mathbf{F}[c]ws(c)$
8. $i : \mathbf{F}[a]ws(a)$
9. $i : \mathbf{F}[b]ws(b)$
10. $w_1 : \mathbf{F}ws(a)$
11. $i\ \rho_a\ w_1$
12. $w_2 : \mathbf{F}ws(b)$
13. $i\ \rho_b\ w_2$
14. $w_3 : \mathbf{F}ws(c)$
15. $i\ \rho_c\ w_3$
16. $w_1\ \rho_\varepsilon\ w_4$
17. $w_2\ \rho_a\ w_4$
18. $w_2\ \rho_\varepsilon\ w_5$
19. $w_3\ \rho_b\ w_5$
20. $i\ \rho_{fool}\ w_2$
21. $w_2\ \rho_{fool}\ w_4$

22. $i\ \rho_{fool}\ w_4$
23. $w_4 : \mathbf{T}(ws(a) \vee ws(b) \vee ws(c))$
24a. $w_4 : \mathbf{T}ws(a)$
$\times$
24b. $w_4 : \mathbf{T}ws(b)$
25b. $w_2 : \mathbf{T}(\neg ws(b) \supset [a]\neg ws(b))$
26ba. $w_2 : \mathbf{F}\neg ws(b)$
27ba. $w_2 : \mathbf{T}ws(b)$
$\times$
26bb. $w_2 : \mathbf{T}[a]\neg ws(b)$
27bb. $w_4 : \mathbf{T}\neg ws(b)$
28bb. $w_4 : \mathbf{F}ws(b)$
$\times$
24c. $w_4 : \mathbf{T}ws(c)$
25c. $i\ \rho_{fool}\ w_3$
26c. $w_3\ \rho_{fool}\ w_5$
27c. $i\ \rho_{fool}\ w_5$
28c. $w_5 : \mathbf{T}(\neg ws(c) \supset [a]\neg ws(c))$
29ca. $w_5 : \mathbf{F}\neg ws(c)$
30ca. $w_5 : \mathbf{T}ws(c)$
31ca. $w_3 : \mathbf{T}(\neg ws(c) \supset [b]\neg ws(c))$
32caa. $w_3 : \mathbf{F}\neg ws(c)$
33caa. $w_3 : \mathbf{T}ws(c)$
$\times$
32cab. $w_3 : \mathbf{T}[b]\neg ws(c)$
33cab. $w_5 : \mathbf{T}\neg ws(c)$
34cab. $w_5 : \mathbf{F}ws(c)$
$\times$
29cb. $w_5 : \mathbf{T}[a]\neg ws(c)$
30cb. $w_4 : \mathbf{T}\neg ws(c)$
31cb. $w_4 : \mathbf{F}ws(c)$
$\times$

We denote with "a", "b", and "c" the three branches which are created by the application of β-rule twice to step 23., "ba" and "bb" the two ones that are created by the β-rule to step 25b., "ca" and "cb" the ones that are created by the β-rule to step 28c. and, finally, "caa" and "cab" the two ones which are created from step 31f. Explanation: *1.*: formula (1) from Example VI.2.1; *2., 3., and 4.*: instances of formula (3) from Example VI.2.1; *5. and 6.*: formulae (4) and (5) from Example VI.2.1; *7.*: the goal; *8.*: from 5., by α-rule; *9.*: from 6., by α-rule; *10. and 11.*: from 8., by π-rule; *12. and 13.*: from 9., by π-rule; *14. and 15.*: from 7., by π-rule; *16. and 17.*: from 11. and 13., by axiom (A_6), when $X = a$ and $Y = b$, and ρ-rule; *18. and 19.*: from 13. and 15., by axiom (A_6), when $X = b$ and $Y = c$, and ρ-rule; *20.*: from 13., by axiom (A_4) and ρ-rule; *21.*: from 17., by axiom (A_3) and ρ-rule; *22.*: from 20. and 21., by axiom (A_2) and ρ-rule; *23.*: from 1. and 22., by ν-rule; *24a., 24b., and 24c.*: from 23., by β-rule, the branch "a" closes due to steps 24a. and 10. since $\overline{w_4} = \overline{w_1}$; *25b.*: from 2. and 20., by ν-rule; *26ba. and 26bb.*: from 25b., by β-rule; *27ba.*: from 26ba., by α-rule, the branch "ba" closes due to 27ba. and 12.;

27bb.: from 26bb. and 17., by ν-rule; *28bb.*: from 27bb., by α-rule, the branch “bb” closes due to 28bb. and 24b.; *25c.*: from 15., by axiom (A_5) and ρ-rule; *26c.*: from 19., by axiom (A_4) and ρ-rule; *27c.*: from 25c. and 26c., by axiom (A_2) and ρ-rule; *28c.*: from 3. and 27c., by ν-rule; *29ca. and 29cb.*: from 28c., by β-rule; *27ca.*: from 29ca., by α-rule, *31ca.*: from 4. and 25c., by ν-rule; *32caa. and 32cab.*: from 31ca., by β-rule; *33caa.*: from 32caa., by α-rule, the branch “caa” closes due to 33caa. and 14.; *33cab.*: from 32cab. and 19., by ν-rule; *34cab.*: from 33cab., by α-rule, the branch “cab” closes due to 34cab. and 30ca; *30cb.*: from 29cb., 17., and 18. (i.e. $w_5\ \rho_a\ w_4$ is available), by ν-rule; *31cb.*: from 30cb., by α-rule, the branch “cb” closes due to 31cb. and 24c.

Remark VI.3.1 Though we have focused on a propositional language, the tableau calculus we have proposed in this chapter can be extended to the first-order case by introducing the rules for quantifiers already seen in Chapter V in the case of the calculus for the class of inclusion modal logics.

Soundness and completeness

In order to prove the soundness and completeness we follow the same guideline of Section III.3. We first prove that the tableau rules preserve the *satisfiability.*

Let $\mathcal{L}$ be a modal language and let $\mathcal{G}$ be a set of incestual axioms. Given a set of prefixed signed formulae and accessibility relation formulae S of $\mathcal{L}$ and a Kripke $\mathcal{G}$-interpretation $M = \langle W, \mathcal{R}_t, V\rangle$, we say $v \in W$ is *$\mathcal{R}_t$-idealizable* if there is some $v' \in W$ such that $(v, v') \in \mathcal{R}_t$. A *$\mathcal{G}$-mapping* is a mapping I from the subset of equivalences classes of the prefixes that occur in some accessibility relation formula of S to W such that if $w\ \rho_t\ w' \in S$ and $I(\overline{w})$ is $\mathcal{R}_t$-idealizable then $(I(\overline{w}), I(\overline{w'})) \in \mathcal{R}_t$. We say S is *$\mathcal{G}$-satisfiable under the $\mathcal{G}$-mapping I in the Kripke $\mathcal{G}$-interpretation M* if, for each $w : \mathbf{T}\varphi$, $M, I(\overline{w}) \models_{\mathcal{G}} \varphi$, for each $w : \mathbf{F}\varphi$, $M, I(\overline{w}) \not\models_{\mathcal{A}} \varphi$, and for each $w\ \rho_t\ w'$, $(I(\overline{w}), I(\overline{w'})) \in \mathcal{R}_t$. Finally, we say a set S of prefixed signed formulae and accessibility relation formulae *$\mathcal{G}$-satisfiable* if S is $\mathcal{G}$-satisfiable under some $\mathcal{G}$-mapping.

A branch of a tableau is $\mathcal{G}$-satisfiable if the set of formulae on it is $\mathcal{G}$-satisfiable and a tableau is $\mathcal{G}$-satisfiable if some its branch is $\mathcal{G}$-satisfiable.

Proposition VI.3.1 *Let T be a $\mathcal{G}$-satisfiable prefixed tableau and let T' be the tableau which is obtained from T by means of one of the extension rules given in Figure VI.3. Then, T' is also $\mathcal{G}$-satisfiable.*

Proof. As in the proof of the Proposition III.3.1, we can focus on application of the extension rules to a branch. The cases when the applied extension rule is the α-rule, β-rule, ν-rule, and π-rule are similar to Proposition III.3.1.

Assume that the applied extension rule is the ρ-rule to obtain S'. Let us suppose $w\ \rho_a\ w'$, and $w\ \rho_c\ w''$ are available in S and that $S' = S \cup \{w'\ \rho_b\ w^*, w''\ \rho_d\ w^*\}$, where $\langle a\rangle[b]\varphi \supset [c]\langle d\rangle\varphi \in \mathcal{G}$ and w^* is new on S. Then, I is already defined for w, w', and w'' and $(I(\overline{w}), I(\overline{w'})) \in \mathcal{R}_a$, $(I(\overline{w}), I(\overline{w''})) \in \mathcal{R}_c$. Since M is a Kripke $\mathcal{G}$-interpretation, by (VI.5), there exist v^* in W such that $(I(\overline{w'}), v^*) \in \mathcal{R}_b$ and $(v^*, I(\overline{w'})) \in \mathcal{R}_d$. This means that $I(\overline{w'})$ is $\mathcal{R}_b$-idealizable and $I(\overline{w''})$ is $\mathcal{R}_d$-idealizable then, we can extend the definition of I by setting $I(w^*) = v^*$.

Assume that the applied extension rule to obtain S' is the ρ_α-rule. Then, an accessibility relation formula of the form $w\ \rho_{t;t'}\ w'$ is in S and $S' = S \cup \{w'\ \rho_t\ w'', w''\ \rho_{t'}\ w'\}$, where $w'' \in \mathcal{W}_C$ is new on S and, therefore, I is not defined on w''. Since $w\ \rho_{t;t'}\ w' \in S$ we have that $(I(\overline{w}), I(\overline{w'})) \in \mathcal{R}_{t;t'}$ and, therefore, there exists a world $v \in W$ such that $(I(\overline{w}), v) \in \mathcal{R}_t$ and $(v, I(\overline{w'})) \in \mathcal{R}_{t'}$. Then, it is enough to extend the definition of I by setting $I(\overline{w''}) = v$.

Assume that the applied extension rule to obtain S' is the ρ_β-rule. Then, an accessibility relation formula of the form $w\ \rho_{t\cup t'}\ w'$ is in S and either $S' = S \cup \{w\ \rho_t\ w'\}$ or $S' = S \cup \{w\ \rho_{t'}\ w'\}$. But, since $w\ \rho_{t;t'}\ w' \in S$, we have that $(I(\overline{w}), I(\overline{w'})) \in \mathcal{R}_{t\cup t'}$ and, therefore, either $(I(\overline{w}), \overline{w'}) \in \mathcal{R}_t$ or $(I(\overline{w}), I(\overline{w'})) \in \mathcal{R}_{t'}$. $\square$

Theorem VI.3.1 (Soundness) *Let $\mathcal{L}$ be a modal language and let $\mathcal{G}$ be a set of incestual axiom schemas, if a formula φ of $\mathcal{L}$ is $\mathcal{T}_{\mathcal{L}}^{\mathcal{G}}$-provable then, it is $\mathcal{G}$-valid.*

Proof. The proof is similar to the one of Theorem III.3.1. $\square$

The completeness is proved by means of the usual counter-model construction. In order to do this we first extend in a suitably way the definition of downward satured set of formulae.

Definition VI.3.4 *Let $\mathcal{L}$, $\mathcal{G}$, and S be a modal language, a set of incestual axiom schemas, and a set of prefixed signed and accessibility relation formulae in $\mathcal{L}$, respectively. Then, we say that S is $\mathcal{G}$-downward satured if:*

1. *for no atomic formula φ, we have $w : \mathbf{T}\varphi \in S$, $w' : \mathbf{F}\varphi \in S$ and $\overline{w} = \overline{w'}$;*
2. *if $w : \alpha \in S$, then $w : \alpha_1 \in S$ and $w : \alpha_2 \in S$;*
3. *if $w : \beta \in S$, then $w : \beta_1 \in S$ or $w : \beta_2 \in S$;*
4. *if $w : \nu^t \in S$, then $w' : \nu_0^t$ is available on S for all w' such that $w\ \rho_t\ w'$ is available on S;*
5. *if $w : \pi^t \in S$, then $w' : \pi_0^t$ is available on S for some w' such that $w\ \rho_t\ w'$ is available on S;*
6. *if $w\ \rho_{t;t'}\ w'$ is available on S then $w\ \rho_t\ w''$ and $w''\ \rho_{t'}\ w'$ are available on S, for some w'';*
7. *if $w\ \rho_{t\cup t'}\ w'$ is in S then $w\ \rho_t w'$ or $w\ \rho_{t'}\ w'$ is available on S;*
8. *if $w\ \rho_a\ w'$ and $w\ \rho_c\ w''$ are available in S and $\langle a\rangle[b]\varphi \supset [c]\langle d\rangle\varphi \in \mathcal{G}$, then $w'\ \rho_b\ w^*$ and $w''\ \rho_d\ w^*$ are available in S, for some w^*.*

Now, we can note that it is quite easy to extend the fair systematic tableau procedure of Figure III.5 for the case of new extension rules presented here, in a such a way to built a $\mathcal{G}$-downward satured set when it produces an open branch.

Definition VI.3.5 (Canonical model) *Given a modal language $\mathcal{L}$, let S be a set of prefixed signed formulae and accessibility relation formulae in $\mathcal{L}$ that is $\mathcal{G}$-downward satured. The canonical model $\mathcal{M}_c^{\mathcal{G}}$ is the ordered triple $\langle W, \mathcal{R}, V\rangle$, where:*

- $W = \{\overline{w} \mid w$ *is used on* $S\}$;
- *for each* $t \in \mathrm{MOD}$, $\mathcal{R}_t = \{(\overline{w}, \overline{w'}) \in W \times W \mid w\ \rho_t\ w'$ *is available on* $S\}$;
- *for each* $p \in \mathrm{VAR}$ *and each* $w \in W$, *we set*

$$V(\overline{w}, p) = \begin{cases} \mathbf{T} & \textit{if } w : \mathbf{T}p \in S \\ \mathbf{F} & \textit{otherwise} \end{cases}$$

Proposition VI.3.2 *Let $\mathcal{M}_c^{\mathcal{G}}$ be the canonical model built by a $\mathcal{G}$-downward satured set of formulae S. Then, $w\ \rho_t\ w'$ is available on S if and only if $(\overline{w}, \overline{w'}) \in \mathcal{R}_t$.*

Proof. The proof is by an easy induction on the structure of the label. (*If* part) If $t = \varepsilon$ and $w\ \rho_t\ w'$ then $\overline{w} = \overline{w'}$ and, therefore, $(\overline{w}, \overline{w'}) \in I$. If $t \in \mathrm{MOD}$ and $w\ \rho_t\ w'$ then $(\overline{w}, \overline{w'}) \in \mathcal{R}_t$ by definition of $\mathcal{M}_c^{\mathcal{G}}$. If $t = t'; t''$ and $w\ \rho_{t';t''}\ w'$ is available on S then, since S is $\mathcal{G}$-downward satured, there are $w\ \rho_{t'}\ w''$ and $w''\ \rho_{t''}\ w'$ available on S, for some w''. By inductive hypothesis, $(\overline{w}, \overline{w''}) \in \mathcal{R}_{t'}$ and $(\overline{w''}, \overline{w'}) \in \mathcal{R}_{t''}$ and, therefore, $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t';t''}$. If $t = t' \cup t''$ and $w\ \rho_{t' \cup t''}\ w'$ is available on S then, since S is $\mathcal{G}$-downward satured, there is $w\ \rho_{t'}\ w'$ or $w\ \rho_{t''}\ w'$ available on S. By inductive hypothesis, $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t'}$ or $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t''}$ and, therefore, $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t' \cup t''}$. (*Only if* part) If $t = \varepsilon$ and $(\overline{w}, \overline{w'}) \in I$ then $\overline{w} = \overline{w'}$ and, therefore, $w\ \rho_\varepsilon\ w'$ is available on S by definition of reflexive, transitive, and symmetric closure of ρ_ε relation. If $t \in \mathrm{MOD}$ and $(\overline{w}, \overline{w'}) \in \mathcal{R}_t$ then $w\ \rho_t\ w'$ is available on S by construction of $\mathcal{M}_c^{\mathcal{G}}$. If $t = t'; t''$ and $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t';t''}$ then $(\overline{w}, \overline{w''}) \in \mathcal{R}_{t'}$ and $(\overline{w''}, \overline{w'}) \in \mathcal{R}_{t''}$, for some $\overline{w''}$. By inductive hypothesis, $w\ \rho_{t'}\ w''$ and $w''\ \rho_{t''}\ w'$ are available on S and, therefore, by definition, $w\ \rho_{t'}\ w'$ is available on S too. Finally, If $t = t' \cup t''$ and $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t';t''}$ then $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t'}$ or $(\overline{w}, \overline{w'}) \in \mathcal{R}_{t''}$. By inductive hypothesis, either $w\ \rho_{t'}\ w''$ or $w''\ \rho_{t''}\ w'$ is available on S and, therefore, by definition, $w\ \rho_{t'}\ w'$ is available on S too. □

Proposition VI.3.3 *The canonical model $\mathcal{M}_c^{\mathcal{G}}$ given by Definition VI.3.5 is a Kripke $\mathcal{G}$-interpretation.*

Proof. We prove that each inclusion properties in $IP_{\mathcal{L}}^{\mathcal{G}}$ is satisfied by $\mathcal{M}_c^{\mathcal{G}}$. Let us suppose that $\mathcal{R}_b \circ \mathcal{R}_d^{-1} \supseteq \mathcal{R}_a^{-1} \circ \mathcal{R}_c \in IP_{\mathcal{L}}^{\mathcal{G}}$, and $(\overline{w}, \overline{w'}) \in \mathcal{R}_a$ and $(\overline{w}, \overline{w'}) \in \mathcal{R}_c$ then, we have to show $(\overline{w'}, \overline{w^*}) \in \mathcal{R}_b$ and $(\overline{w''}, \overline{w^*}) \in \mathcal{R}_d$. If $(\overline{w}, \overline{w'}) \in \mathcal{R}_a$ and $(\overline{w}, \overline{w'}) \in \mathcal{R}_c$ then, by Proposition VI.3.2, $w\ \rho_a\ w'$ and $w\ \rho_c\ w''$ are available on S. Now, since by hypothesis S is $\mathcal{G}$-downward satured, by point (8) of Definition VI.3.4, $w'\ \rho_b\ w^*$ and $w''\ \rho_d\ w^*$ are available on S, for some w^*. Thus, by Proposition VI.3.2, $(\overline{w'}, \overline{w^*}) \in \mathcal{R}_b$ and $(\overline{w''}, \overline{w^*}) \in \mathcal{R}_d$. □

Now, we can prove the key lemma (the model existence) to proving the completeness.

Lemma VI.3.1 *Given a modal language $\mathcal{L}$, if S is a set of prefixed signed formulae and accessibility relation formulae of $\mathcal{L}$ that is $\mathcal{G}$-downward satured then, S is $\mathcal{G}$-satisfiable.*

Proof. Suppose S is $\mathcal{G}$-downward satured. For every formula φ and every prefix w, we have that if $w : \mathbf{T}\varphi \in S$ then $\mathcal{M}_c^{\mathcal{G}}, \overline{w} \models_{\mathcal{G}} \varphi$ and if $w : \mathbf{F}\varphi \in S$ then $\mathcal{M}_c^{\mathcal{G}}, \overline{w} \not\models_{\mathcal{G}} \varphi$. That is, the mapping $I(w) = \overline{w}$ is an $\mathcal{G}$-mapping for S in the Kripke $\mathcal{A}$-interpretation $\mathcal{M}_c^{\mathcal{G}}$. The proof is by induction on the structure of φ. The case of formulae of type α and β are trivial. Let us suppose $w : \nu^t \in S$ then, since S is $\mathcal{G}$-downward satured, $w' : \nu_0^t \in S$ for all w' such that $w\ \rho_t\ w'$ is available on S. By inductive hypothesis, we have that $\mathcal{M}_c^{\mathcal{G}}, \overline{w'} \models_{\mathcal{G}} \nu_0^t$, for each world $\overline{w'}$ such that $(\overline{w}, \overline{w'}) \in \mathcal{R}_t$, hence, by definition of satisfiable relation, $\mathcal{M}_c^{\mathcal{G}}, \overline{w} \models_{\mathcal{G}} \nu^t$. Let us assume, now, $w : \pi^t \in S$ then, since S is $\mathcal{G}$-downward satured, $w' : \pi_0^t \in S$ for some w' such that $w\ \rho_t\ w'$ is available on S. By inductive hypothesis, we have that $\mathcal{M}_c^{\mathcal{G}}, \overline{w'} \models_{\mathcal{G}} \pi_0^t$, for some world $\overline{w'}$ such that $(\overline{w}, \overline{w'}) \in \mathcal{R}_t$, hence, by definition of satisfiable relation, $\mathcal{M}_c^{\mathcal{G}}, \overline{w} \models_{\mathcal{G}} \pi^t$. □

Theorem VI.3.2 (Completeness) *Let $\mathcal{L}$ be a modal language and let $\mathcal{G}$ be a set of incestual axiom schemas, if a formula φ of $\mathcal{L}$ is $\mathcal{G}$-valid then, φ is $\mathcal{T}_{\mathcal{L}}^{\mathcal{G}}$-provable.*

Proof. The proof is similar to the one of Theorem III.3.2. □

Chapter VII

Related work

In this part of the thesis, we have presented the class of *inclusion modal logics*. This class includes some well-known modal systems such as K_n, T_n, $K4_n$, $S4_n$. However, differently than other proposals, these systems can be *non-homogeneous* and can contain arbitrarily complex *interaction axioms*: features particularly suitable for modal systems modeling, for instance, knowledge and beliefs in multiagent situation.

An analytic tableau calculus for this class of logics has been developed. In order to have a general framework able to cope with any kind of inclusion axioms, we have chosen the simplest way of representing models: prefixes are worlds, and relations between them are built step by step by the rules of the calculus. In particular, axioms are used as rewrite rules which create new paths among worlds.

The calculus is then extended in order to deal with the class of incestual modal logics as defined in [Catach, 1988]. This allows to deal also with multimodal logics characterized, among other things, by *serial*, *symmetric*, and *Euclidean* accessibility relations. Furthermore, some (un)decidability results for the class of inclusion modal logics are given.

VII.1 Prefixed tableau systems

Our approach to prefixed tableaux and, in particular, to represent accessibility relations by means of a graph is closely related to the approaches based on prefixes used in [Fitting, 1983] and by other authors for classical modal (though no multimodal) systems [Massacci, 1994; Goré, 1995] and for dynamic logic [De Giacomo and Massacci, 1996]. In these works, prefixes are sequences of integers which represent a world as a path in the model, that connects the initial world to the one at hand. Thus, instead of representing a model as a graph, as in the our approach, a model is represented as a set of paths which can be considered a spanning tree of the same graph. Although this representation may be more efficient, the disadvantage is that it requires a specific ν-rule for each logic. These rules code the properties of accessibility relations. Depending on the logic, the ν-rules may express complex relations between prefixes, which instead in our case are explicitly available from the representation. In particular, Massacci has proposed a "single step calculus" where ν-rules make use only of immediately accessible prefixes [Massacci, 1994]. His approach works for all the distinct basic normal logics obtainable from K by addition on any combination of the axiom T, D, 4, 5, and B in a modular way but it still requires the definition of specific ν-rules. On the contrary, our calculus deals with all modal logic considered by [Fitting, 1983; Massacci, 1994; Goré, 1995] and many others by means of the only ρ-rule. Moreover, it is *modular* with respect to the characterizing axioms of the multimodal logic, i.e., it is enough to know the axioms to get the calculus.

Besides the disadvantage of requiring specific ν-rules and the fact that they do not work with multimodal systems, we think that it is difficult to extend the approach based on prefixes as sequences to the whole class even though it might be adapted for some subclasses of inclusion and incestual axioms. In particular, it can be shown that a "generation lemma" ([Massacci, 1994, page 732] [Goré, 1995, Section 6.2]) does not hold, i.e. it is not true that, for any prefix occurring on a branch, all intermediate prefixes occur too. This property is at the basis of the completeness proof for the calculus in [Massacci, 1994; Goré, 1995]. Let us consider the following example.

Example VII.1.1 Assume that the multimodal logic $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$ is characterized by the inclusion axiom $[a][b]\varphi \supset [c]\varphi$. Then, the formula $[a]p \wedge \langle c\rangle q \supset \langle a\rangle p$ is provable:

1.	$i : \mathbf{F}([a]p \wedge \langle c\rangle q \supset \langle a\rangle p)$
2.	$i : \mathbf{T}[a]p \wedge \langle c\rangle q$
3.	$i : \mathbf{F}\langle a\rangle p$
4.	$i : \mathbf{T}[a]p$
5.	$i : \mathbf{T}\langle c\rangle q$
6.	$w_1 : \mathbf{T}q$
7.	$i \; \rho_c \; w_1$
8.	$i \; \rho_a \; w_2$
9.	$w_2 \; \rho_b \; w_1$
10.	$w_2 : \mathbf{F}p$
11.	$w_2 : \mathbf{T}p$
	$\times$

Explanation: *1.*: the goal; *2. and 3.*: from 1., by α-rule; *4. and 5.*: from 2., by α-rule; *6. and 7.*: from 5., by π-rule; *8. and 9.*: form 7., by ρ-rule from axiom $[a][b]\varphi \supset [c]\varphi$; *10.*: from 3. and 8., by ν-rule; *11.*: from 4. and 8., by ν-rule; The branch close due to steps 10. and 11.

By applying π-rule to the prefixed formula at step 5., we get a new world w_1 (step 6. and step 7.). We can imaging to use the prefix "1.1_c" to represent the world w_1 (see Figure VII.1):

1.	$1. : \mathbf{F}([a]p \wedge \langle c\rangle q \supset \langle a\rangle p)$
2.	$1. : \mathbf{T}[a]p \wedge \langle c\rangle q$
3.	$1. : \mathbf{F}\langle a\rangle p$
4.	$1. : \mathbf{T}[a]p$
5.	$1. : \mathbf{T}\langle c\rangle q$
6.	$1.1_c. : \mathbf{T}q$

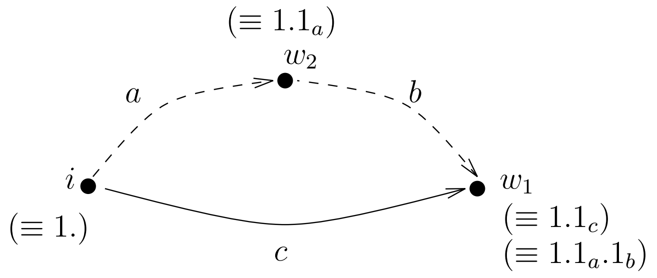

Figure VII.1: ρ-rule as rewriting rule: counter-model construction of Example VII.1.1.

Now, by applying axiom $[a][b]\varphi \supset [c]\varphi$, the same world can also be represented by the sequence "$1.1_a.1_b$" (accessibility relation formulae at steps 8. and 9. in Example VII.1.1):

6. $1.1_a.1_b. : \mathbf{T}q$

whose subprefix "1.1_a" (world w_2 in Figure VII.1) does not occur on the branch. On the other hand, this subprefix (world) is needed to apply the ν-rule to the formula at step 3. and 4. in order to close branch.

Moreover, adding explicitly subprefixes, as the one above, is not enough to solve the problem, since all prefixes representing the same world have to be identified.

Example VII.1.2 Assume that the multimodal logic $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}}$ is characterized by the inclusion axioms $[a]\varphi \supset [c]\varphi$ and $[b]\varphi \supset [c]\varphi$. Then, the formula $[a]p \wedge \langle c\rangle q \supset \langle b\rangle p$ is provable:

1. $i : \mathbf{F}([a]p \wedge \langle c\rangle q \supset \langle b\rangle p)$
2. $i : \mathbf{T}[a]p \wedge \langle c\rangle q$
3. $i : \mathbf{F}\langle b\rangle p$
4. $i : \mathbf{T}[a]p$
5. $i : \mathbf{T}\langle c\rangle q$
6. $w_1 : \mathbf{T}q$
7. $i\ \rho_c\ w_1$
8. $i\ \rho_a\ w_1$
9. $i\ \rho_b\ w_1$
10. $w_1 : \mathbf{F}p$
11. $w_1 : \mathbf{T}p$

$\times$

Explanation: *1.*: the goal; *2. and 3.*: from 1., by α-rule; *4. and 5.*: from 2., by α-rule; *6. and 7.*: from 5., by π-rule; *8.*: form 7., by ρ-rule from axiom $[a]\varphi \supset [c]\varphi$; *9.*: form 7., by ρ-rule from axiom $[b]\varphi \supset [c]\varphi$; *10.*: from 3. and 9., by ν-rule; *11.*: from 4. and 8., by ν-rule; The branch close due to steps 10. and 11.

Using prefixes *á la* Fitting we can represent the world w_1 by means of the prefix 1.1_c, that is:

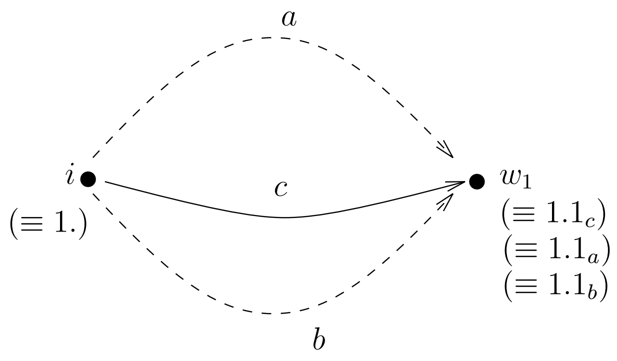

Figure VII.2: ρ-rule as rewriting rule: counter-model construction of Example VII.1.2.

1.	$1. : \mathbf{F}([a]p \wedge \langle c\rangle q \supset \langle b\rangle p)$
2.	$1. : \mathbf{T}[a]p \wedge \langle c\rangle q$
3.	$1. : \mathbf{F}\langle b\rangle p$
4.	$1. : \mathbf{T}[a]p$
5.	$1. : \mathbf{T}\langle c\rangle q$
6.	$1.1_c. : \mathbf{T}q$

Now, by applying axiom $[a]\varphi \supset [c]\varphi$ and axiom $[b]\varphi \supset [c]\varphi$ the same world w_2 will be denoted by the prefixes 1.1_a and 1.1_b:

7.	$1.1_a : \mathbf{T}q$
8.	$1.1_b : \mathbf{T}q$

and, then, applying twice the ν-rule to the formulae at steps 3. and 4. we have:

9.	$1.1_b : \mathbf{F}p$
10.	$1.1_a : \mathbf{T}p$

but the branch does not close because we cannot identify 1.1_b and 1.1_a which are the same world (see Figure VII.2), whereas our calculus does (see Example VII.1.2).

Other tableau methods for propositional modal logics which make use of prefixed formulae are presented in [Governatori, 1995; Cunningham and Pitt, 1996]. The system in [Cunningham and Pitt, 1996] deals with all the fifteen propositional normal modal logics obtained by combining the axioms T, D, 4, 5, and B, while the system in [Governatori, 1995] considers the propositional modal logics $K45$, $D45$, and $S5$ and the propositional modal logics $S5A$ and $S5P_{(n)}$. It has subsequently been extended to deal with the above mentioned fifteen modal systems and the predicative case in [Artosi *et al.*, 1996; Governatori, 1997]. These proof systems extend the calculus KE, a combination of tableau and natural deduction inference rules which allows for a suitably restricted use of the cut rule [D'Agostino and Modadori, 1994]. In order to have a more efficient proof search, they generalize the prefix both allowing the occurrence of variables and using unification to show that two prefixes can name the same world. The main difference between the system in [Governatori, 1995; Artosi *et al.*, 1996; Governatori, 1997] and the one in [Cunningham and Pitt, 1996] is that the former uses only one type of path variable (single worlds) while the latter allows variables over single as well as sequences of worlds. Furthermore, in [Governatori, 1995], only one ν-rule is used and unification is logic-dependent while, in [Cunningham and Pitt, 1996], unification is independent of the logic but there is a different ν-rule for each logic.

One of the main features of these systems is the full permutability of the application of their rules. Unfortunately, our tableau method does not enjoy this property. In fact, similarly to the problem of applying the existential rules before the universal ones in the proof systems for classical logic, we need to apply the π-rules (or the ρ-rules) before the ν-rules. On the other hand, we deal with a wider class of logics. In particular, we think that it is hard to extend the unification method of prefixes so to deal with all the classes of logics that we considered for the same reasons given above in the case of classical prefixed systems.

In [Catach, 1991] a general theorem prover for propositional modal logics is presented. This system, named TABLEAUX, uses a representation for the accessibility relations that is close to ours. In fact, in that work a tableau is a pair (Γ, R), where Γ is a set of prefixed formulae and R is a set of relations between worlds. Prefixes are constant symbols.

TABLEAUX can deal with all the already mentioned fifteen modal systems, and also with their multimodal versions. However, it does not deal with any interaction axiom while our does. This system uses three classes of tableau rules: the first is made of *simplification* rules, that are world independent and whose aim is to simplify the proof search; the second consists of the *transformation* rules and allows to introduce new operators in terms of the existing ones; finally, the third class of rules deals with formulae belonging to different worlds and can introduce modifications in the set R of relations.

VII.2 Translation methods

Instead of developing specific theorem proving techniques and tools for modal logics, many authors have proposed the alternative approach of translating modal logics into classical first order logic, so that standard theorem provers can be used without the need to built new ones [Ohlbach, 1993b]. The translation methods are based on the idea of making explicit reference to the worlds by adding to all predicates an argument representing the world where the predicate holds, so that the modal operators can be transformed into quantifiers of classical logic.

The *relational translation* is based on the direct simulation of the Kripke semantics by introducing a distinguished predicate symbol to represent the accessibility relation [Moore, 1980]. This method has strong relationships with our approach. Indeed, we deal with inclusion properties of the accessibility relations, which are first-order axiomatizable, hence, the relational translation method can cope with them. On the other hand, as a drawback, the relational translation method destroys the structure of the formulae and it may cause an exponential growth of translated formulae.

An alternative method is the *functional translation* [Ohlbach, 1991; Auffray and Enjalbert, 1992]. It is based on the idea of representing paths in the possible worlds structure by means of compositions of functions, which map worlds to accessible worlds. The most common properties, such as transitivity or reflexivity, are taken into account by an equational unification algorithm. An advantage of this approach is that it keeps the structure of the original formula.

In [Ohlbach, 1993a; Gasquet, 1993] various optimizations of the functional translation method are investigated. In particular, a substantial simplification can be obtained for the case that all accessibility relations are *serial*. However, even in this case equational unification cannot be avoided. In particular, an optimization method for the class of inclusion logics has been presented in [Gasquet, 1993]. Gasquet shows that it is possible to get rid of the *sort* denoting possible worlds, used in [Ohlbach, 1991], when we deal with inclusion modal logics. Nevertheless, the *seriality* is assumed for each accessibility relation and, hence, this approach cannot be adopted, for instance, to deal with the logic we have introduced in Example III.2.3 at page 26.

A way to avoid the use of equational unification algorithms, retaining the advantages of the functional translation, has been developed in [Nonnengart, 1993], where a mixed approach based on a relational and functional translation is defined. One of the aims of the author was to obtain Prolog programs starting from Horn clauses extended with modal operators [Nonnengart, 1994]. This method requires that accessibility relation properties are first-order predicate logic definable. In particular, he can provide a translation for the modal systems (all requiring seriality) KD, KT, $KD4$, $S4$, but he can deal also with axioms like $(B) : \varphi \supset \Box\Diamond\varphi$, and, then, with logics like KDB, $KD45$, $S5$ and the multimodal system $KD45_n$.

Part Two

Inclusion Modal Logics for Programming

Chapter VIII

Introduction

The problem of extending logic programming languages with modal operators has raised a lot of attention in the last years. Several researchers have proposed extensions of logic programming with temporal logics and with modal logics (see [Orgun and Ma, 1994; Fisher and Owens, 1993b] for detailed overviews) providing tools for formalizing temporal and epistemic knowledge and reasoning, that retain the characterizing properties of logic programming languages, such as, for instance, *goal directed proof procedures*, *fixed point semantics* and the notion of *minimal Herbrand model*.

In this part of the thesis, we define a *logic programming language*, called NemoLOG (which stands for New modal proLOG), that is based on the class of *first-order inclusion modal logics* introduced in the previous part. It extends the language of *Horn clauses* with modal operators which, in particular, can occur *in front of clauses*, *in front of clause heads* and *in front of goals*.

NemoLOG is *parametric* with respect to the properties of modal operators determined by means of the set of *inclusion axiom schemas* which, in turn, determine the underlying inclusion modal logic. We show that this extension is well suited for *structuring knowledge* and, in particular, for defining *module constructs* within programs, for representing *agents beliefs* and performing *epistemic* reasoning, simple forms of reasoning about *actions*, and for interpreting some features of *object-oriented* paradigms in logic programming, such as *hierarchical* dependencies and *inheritance* among classes.

One of the aims in defining NemoLOG comes from the need of defining *structuring facilities* to enhance *modularity*, *readability*, and *reusability* of logic programs. Logic languages use flat collections of Horn clauses and they lack mechanisms for structuring programs, which are instead available in other programming paradigms. This problem has attracted a lot of interest and many different approaches have been proposed (see [Bugliesi *et al.*, 1994] for a detailed survey). In this thesis, in the line of some previous languages, such as those defined in [Baldoni *et al.*, 1993; Giordano and Martelli, 1994; Baldoni *et al.*, 1997a], we address this topic by means of the modal logic, using universal modal operators to define *modules*. The key idea is to associate a modal operator with each module in order to label its clauses. Module composition is obtained by allowing modules to export clauses or derived facts. To achieve this purpose, we use again a modal operator which makes it possible to distinguish among clauses local to module, clauses that are fully exported from a module, and those whose consequences only are exported. As we will see, NemoLOG allows to model different kinds of modules presented in the literature (such as [Monteiro and Porto, 1989; Brogi *et al.*, 1990a; Brogi *et al.*, 1990b]).

Another important problem related to providing support for *software engineering* is the integration of logic programming and *object-oriented paradigms* [Turini, 1995]. A significant proposal to tackle this problem is the one by McCabe in [McCabe, 1992], where the idea of representing an object as a first-order logic theory is exploited. From a different perspective, in this thesis, we show how modal logics and, in particular, inclusion modal logics can be used to interpret the object-oriented paradigms in logic programming. *Hierarchical dependencies* among modules (classes) can be represented by means of nested modules or by inclusion axiom schemas. For example, if $[m_1]M_1$ and $[m_2]M_2$ represent two modules, where M_1 and M_2 are sets of clauses, the inclusion axiom

$$[m_1]\varphi \supset [m_2]\varphi$$

says that all the clauses of module m_1 are *exportable* into module m_2; in different words m_1 is a more specific class of m_2. Besides, a behaviour similar to the use of *self* can be obtained by means of a modal operator which is a sort of common knowledge operator.

In Chapter IX, a *goal directed proof procedure*, which is *modular* with respect to the chosen set of inclusion axiom clauses, is presented by making use of a notion of *derivation relation* between sequences of modal operators. The derivation relation *only* depends on the properties of modalities themselves (i.e., it is based on the set of inclusion axiom clauses contained in the program). More specifically, the proof procedure is based on a notion of *modal context*, where modal context is a sequence of modal operators, which keeps trace of the ordering between modalities found in front of goals during a computation so that a modal context is associated with each goal to be solved. According to the modal context in which a subgoal has to be proved, a given clause of the program may or may not be used to solve it, depending both on the modal structure of the clause itself and on its "relation" to the modal context of the goal. This relation is defined by the above mentioned derivation relation; thus, the derivation relation is used to *select* a clause for proving a goal in a certain modal context, *according to* the properties of modalities of the clause. These properties are completely specified by the derivation relation, that can be regarded as a *rewriting system* [Book, 1987]. The sequences of modalities are the domain of the strings and the rewriting rules are the axioms characterizing the modal operators of the underlying logic (specified by means of the inclusion axiom clauses).

In this part of the thesis, we also investigate the relationship between NemoLOG and the general proof theory presented in Chapter III. In particular, we, first, introduce a sequent calculus that is a simple syntactical transformation of our tableau method and, then, we prove that, in the case of NemoLOG, we can restrict our attention to sequent proofs of a form, that corresponds to the *uniform proofs* in the meaning of [Miller *et al.*, 1991]. This kind of proofs have a lot of importance because they can be constructed in a goal-directed manner and, thus, automated deduction based on this kind of proofs can be optimized.

This result is achieved due to the more "flexibility" of all prefixed tableau methods in the application of the rules during the construction of a proof.

We show that our goal directed proof procedure is *sound* and *complete* with respect to the possible-worlds semantics presented in Chapter V. To do this we define a *fixed point* semantics by *generalizing* the standard construction of Horn clauses and we prove its completeness with respect to the possible-worlds semantics through a *canonical model* construction. Though the construction is pretty standard, we believe that its advantage is in the modularity of the approach, i.e., both the completeness and soundness proof are *modular* with respect to the underlying inclusion modal logics of the programs and so they work for the whole class of inclusion modal systems.

This part of the thesis is organized as follows. NemoLOG is introduced in Chapter IX. The *operational* semantics is presented and some examples of *programs* and *operational derivations* are discussed. Moreover, the relations with the general proof theory of the inclusion modal logics is shown. In Chapter X, we show some interesting applications of the defined modal extension of Horn clauses, while in Chapter XI, we define the *fixed point* semantics and we give the proof of soundness and completeness of operational semantics with respect to possible-worlds semantics. Finally, in Chapter XII, we overview some related works. They are divided in two classes: the ones that are based on inclusion modal logics and the ones that are not.

Chapter IX

A Programming Language

In this chapter we introduce NemoLOG, our modal logic programming language. It extends *Horn clause logic* allowing modalities to occur in clauses and in goals. In particular, it allows free occurrences of some *universal modalities* of the form $[t]$, where t is an arbitrary term of the language, *in front of* clauses, clause heads and goals. A *goal directed proof procedure* will be defined and, at the end, we will investigate the relationship between programs and goals of NemoLOG and the tableau methods studied in the first part of the thesis. Finally, we give a method for translating NemoLOG programs into standard Horn clause logic, so that the translated programs can be executed by any Prolog interpreter or compiler.

IX.1 Syntax

Given a first-order modal language $\mathcal{L}_{FO}$ (see page 46) we define NemoLOG as a *first-order modal logic programming language* whose *alphabet* contains:

- all the symbols of $\mathcal{L}_{FO}$ apart from the classical connectives "$\vee$", and "$\neg$";
- the distinguished symbol T (*true*);
- the *binary operator* "$\rightarrow$";
- the symbol "ε" denoting the *empty sequence* of modalities.

Definition IX.1.1 (Modalized goals) *The set* GOAL *of* modalized goals *in* NemoLOG *is defined as the least set of formulae that satisfies the following conditions:*

- $T \in$ GOAL*;*
- *if A is an* atomic formulae *of* FOR *then,* $A \in$ GOAL*;*
- *if* $G_1, G_2 \in$ GOAL *then,* $G_1 \wedge G_2 \in$ GOAL*;*

- *if* $G \in \text{GOAL}$ *and* $x \in \text{VAR}$ *then,* $\exists x G \in \text{GOAL}$*;*
- *if* $t \in \text{TERM}$ *and* $G \in \text{GOAL}$ *then,* $[t]G \in \text{GOAL}$*.*

Definition IX.1.2 (Modalized defined clauses) *The set* DEFC *of* modalized defined clauses *in* NemoLOG *is defined as the least set of formulae that satisfies the following conditions:*

- *if* $G \in \text{GOAL}$*,* A *is an atomic formulae of* FOR*, and* Γ *is a sequence of modalities*[1] *(possible empty) then,* $G \supset \Gamma A \in \text{DEFC}$*,* ΓA *is named* modalized clause head*;*
- *if* $D_1, D_2 \in \text{DEFC}$ *then* $D_1 \wedge D_2 \in \text{DEFC}$*;*
- *if* $t \in \text{TERM}$ *and* $D \in \text{DEFC}$ *then,* $[t]D \in \text{DEFC}$*;*
- *if* $D \in \text{DEFC}$ *and* $x \in \text{VAR}$ *then,* $\forall x D \in \text{DEFC}$*.*

NemoLOG allows free occurrence of modal operators *in front of clauses*

$$[t_1][[t_2](a \wedge b \supset c),$$

in front of clause heads

$$[t_1][[t_2](a \wedge b \supset [t_3][t_4]c),$$

and *in front of each goal*

$$[t_1][[t_2]([t_5]a \wedge [t_6][t_7]b \supset [t_3][t_4]c).$$

Definition IX.1.3 (Inclusion axiom clauses) *The set* INC *of* inclusion axiom clauses *in* NemoLOG *is defined as the least set of formulae that satisfies the following condition:*

- *if* Γ_1 *is a non-empty sequence of modalities and* Γ_2 *is a possible empty sequence of modalities*[2] *then,* $\Gamma_1 \rightarrow \Gamma_2 \in \text{INC}$*.*

We will refer to modalized clauses, modalized goals, modalized clause heads, and inclusion axiom clauses with clauses, goals, clause heads and axiom clauses when no confusion arises.

Definition IX.1.4 (Program) *A program* P *in* NemoLOG *is a* pair $\langle Ds, Ax\rangle$*, where:*

- Ds *is a set of modalized defined clauses of* DEFC*; and*
- Ax *is a finite (possible empty) set of inclusion axiom clauses of* INC*.*

Intuitively, assume that NemoLOG is based on the first-order modal language $\mathcal{L}_{FO}$ and let $\langle Ds, Ax\rangle$ be a program of NemoLOG. Then, the set Ds of clauses can be considered the actual program specification, while the set Ax of axiom clauses represents the set of inclusion axiom schemas the characterizes the *underlying* inclusion modal logic of the program. More precisely, the underlying logic of the set of clauses Ds is $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$, where $\mathcal{A} = \{[t_1]\dots[t_n]\varphi \supset [s_1]\dots[s_m]\varphi \mid [t_1]\dots[t_n] \rightarrow [s_1]\dots[s_m] \in Ax\}$.

[1]For instance, Γ could be $[t_1][t_2]\dots[t_n]$.

[2]Denoted by "ε".

Some examples of modal logic programs

To give an idea of how a program in NemoLOG is defined, let us consider two simple examples. The former is a formulation of the *Fibonacci* example from [Abadi and Manna, 1989], while the latter presents the *friends puzzle* of Example II.3.3.

Example IX.1.1 *(The Fibonacci numbers)* In this example the modal operator $[next]$ represents the next instant of time and it is axiomatized only by the axiom K, while $[always]$ denote a temporal operator used to represent something that holds in any instant of time. $[always]$ is axiomatized by the following:

(A_1) $T(always) : [always]\varphi \supset \varphi;$
(A_2) $4(always) : [always]\varphi \supset [always][always]\varphi;$
(A_3) $I(always, next) : [always]\varphi \supset [next]\varphi.$

We want $fib(X)$ to hold after n instants of time, if X is equal to Fibonacci of n. The formulation is given by Program IX.1.

Program IX.1 : Fibonacci numbers.

Ax: (1) $[always] \rightarrow \varepsilon$
(2) $[always] \rightarrow [always][always]$
(3) $[always] \rightarrow [next]$

Ds: (4) $T \supset fib(0)$
(5) $T \supset [next]fib(1)$
(6) $\forall X \forall Y \forall Z([always](fib(Y) \wedge [next]fib(Z) \wedge X\ is\ Y + Z \supset$
$[next][next]fib(X)))$

Axiom clauses (1), (2), and (3) represent the inclusion modal axioms (A_1), (A_2), and (A_3), respectively. Clause (4) says that at time 0, $fib(0)$ holds; clause (5) says that at time 1, $fib(1)$ holds; clause (6) says that, for any time n, if $fib(Y)$ holds at time n, and if $fib(Z)$ holds at time $n+1$, then $fib(X)$, with $X = Y + Z$, holds at time $n+2$. The sequence $[next] \ldots [next]$ of $n \geq 0$ modalities is used to represent what holds after n instants of time. From this program, the query $[next][next][next]fib(X)$ succeeds with $X = 2$, and indeed 2 is Fibonacci of 3.

Example IX.1.2 *(The friends puzzle)* The Program IX.2 shows the NemoLOG version of Example II.3.3.

Program IX.2 : Friends puzzle.

Ax: (1) $[peter][john] \rightarrow [john][peter]$
(2) $[peter] \rightarrow \varepsilon$
(3) $[peter] \rightarrow [peter][peter]$
(4) $[john] \rightarrow \varepsilon$
(5) $[john] \rightarrow [john][john]$
(6) $[wife(peter)] \rightarrow [peter]$
(7) $[wife(peter)] \rightarrow \varepsilon$

(8) $[wife(peter)] \rightarrow [wife(peter)][wife(peter)]$

Ds: (9) $[peter]time$
(10) $[wife(peter)]([peter]time \supset [john]time)$
(11) $[peter][john]place$
(12) $[peter][john](place \wedge time \supset appointment)$

Again, the set Ax represents the inclusion axioms of the underlying modal logic of the set of clauses Ds (see axioms (A_1)-(A_8) of Example II.3.3). The goal

$$[john][peter]appointment \wedge [peter][john]appointment$$

succeeds from the program $\langle Ds, Ax\rangle$.

IX.2 Operational semantics

In this section we introduce a *goal directed proof procedure* for our modal logic programming language but, before to do this, we need to give some more notions.

Derivability relation

Since modalities are allowed to occur freely in front of goals, when proving a goal G from a program P we need to record the sequence of modalities which occur in the goal, that is the *modal context* in which each subgoal has to be proved. According to the modal context in which a subgoal has to be proved, a given clause of the program may be used or not to solve it: it depends on the modal structure of the clause itself, and on its *relation* to the modal context of the goal (see also [Baldoni *et al.*, 1993; Giordano and Martelli, 1994; Baldoni *et al.*, 1997a]). For instance, given a goal $[t_1][t_2]p$, the sequence $[t_1][t_2]$ represents the modal context for the goal p. Assume that the program contains a clause $[t_3]p$. This clause can be used to solve the goal p only if the modality $[t_3]$ relates somehow to the context $[t_1][t_2]$. For instance, if our set Ax of inclusion axiom clauses contains the axiom clause $[t_3] \rightarrow [t_1][t_2]$ (that is, the underlying logic is characterized by axiom schema $[t_3]\varphi \supset [t_1][t_2]\varphi$), then the clause $[t_3]p$ can certainly be used to prove the goal.

We formalize this relationship between sequences of modalities (the modalities in the clause and the modalities in the modal context of a goal) by introducing a *derivation relation* between them. This relation will depend on the inclusion axiom clauses in Ax of the program (and, therefore, by the logical axioms $\mathcal{A}$ of the underlying logic).

More formally, let $\mathcal{C}$ be a set of all *ground* modal operators of the form $[t]$, where t is a ground term of a language NemoLOG. We define the set of modal contexts $\mathcal{C}^*$ as the set of all finite sequences on $\mathcal{C}$, including the empty sequence "ε". Moreover, we denote with $[Ax]$ the set of all ground instance of the axiom clauses in Ax.

Definition IX.2.1 (Derivation relation) *Given a set Ax of inclusion axiom clauses, the* derivation relation $\overset{*}{\Rightarrow}_{Ax}$ generated by *Ax is the the* transitive *and* reflexive *closure of* *the relation* $\Rightarrow_{Ax}$ *defined as follows: for each* $\Gamma_1 \rightarrow \Gamma_2 \in [Ax]$ *and* $\Gamma, \Gamma' \in \mathcal{C}^*$, $\Gamma\Gamma_1\Gamma' \Rightarrow_{Ax} \Gamma\Gamma_2\Gamma'$.[3]

Given a set Ax of axiom clauses two sequences of modalities Γ_1 and Γ_2, we say that Γ_1 *derives* Γ_2 if $\Gamma_1 \overset{*}{\Rightarrow}_{Ax} \Gamma_2$; in this case Γ_1 is an *ancestor* of Γ_2 and Γ_2 is a *descendant* of Γ_1. We can prove the following property.

Proposition IX.2.1 *Given a set of inclusion axiom clauses* Ax*, for all formula* ψ *of* $\mathcal{L}_{FO}$ *and for all* $\Gamma, \Gamma' \in \mathcal{C}^*$*, if* $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$ *then* $\models_{\mathcal{A}} \Gamma\psi \supset \Gamma'\psi$*, where* $\mathcal{A} = \{\Gamma_1\varphi \supset \Gamma_2\varphi \mid \Gamma_1 \rightarrow \Gamma_2 \in Ax\}$.

Proof. The proof is by induction on the definition of $\overset{*}{\Rightarrow}_{Ax}$. *(Base)* If $\Gamma\Gamma_1\Gamma' \overset{*}{\Rightarrow}_{Ax} \Gamma\Gamma_2\Gamma'$ and $\Gamma_1 \rightarrow \Gamma_2 \in [Ax]$, then we have to prove $\models_{\mathcal{A}} \Gamma\Gamma_1\Gamma\psi \supset \Gamma\Gamma_2\Gamma'\psi$, that is for all Kripke $\mathcal{A}$-interpretation M and all world w in W, we have $M, w \models_{\mathcal{A}} \Gamma\Gamma_1\Gamma\psi \supset \Gamma\Gamma_2\Gamma'\psi$. Let us assume $M, w \models_{\mathcal{A}} \Gamma\Gamma_1\Gamma\psi$ and prove $M, w \models_{\mathcal{A}} \Gamma\Gamma_2\Gamma'\psi$. If $M, w \models_{\mathcal{A}} \Gamma\Gamma_1\Gamma\psi$ then, for any sequence of worlds $w_1, \ldots, w_n$, such that $(w, w_1) \in \mathcal{R}_{V(t_1)}, \ldots, (w_{n-1}, w_n) \in \mathcal{R}_{V(t_n)}$, where $[t_1] \ldots [t_n]$ is Γ, we have that $M, w_n \models_{\mathcal{A}} \Gamma_1\Gamma\psi$. Now, $\models_{\mathcal{A}} \Gamma_1\varphi \supset \Gamma_2\varphi$, for any formula φ of $\mathcal{L}_{FO}$ and, in particular, $\models_{\mathcal{A}} \Gamma_1(\Gamma'\psi) \supset \Gamma_2(\Gamma'\psi)$. Thus, since $M, w_n \models_{\mathcal{A}} \Gamma_1(\Gamma'\psi)$, we have $M, w_n \models_{\mathcal{A}} \Gamma_2(\Gamma'\psi)$, for any sequence of worlds $w_1, \ldots, w_n$, that is, $M, w \models_{\mathcal{A}} \Gamma\Gamma_2\Gamma'\psi$. *(Reflexivity)* The case of reflexivity closure is trivial. *(Transitivity)* Let us assume that $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$ and $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma''$ and $\Gamma'' \overset{*}{\Rightarrow}_{Ax} \Gamma'$, we have to prove $\models_{\mathcal{A}} \Gamma\psi \supset \Gamma'\psi$. By inductive hypothesis $\models_{\mathcal{A}} \Gamma\psi' \supset \Gamma''\psi'$ and $\models_{\mathcal{A}} \Gamma''\psi'' \supset \Gamma'\psi''$, for any formula ψ' and ψ'' of $\mathcal{L}_{FO}$ and, in particular, for $\psi' = \psi$ and $\psi'' = \psi$. Let us assume that $\models_{\mathcal{A}} \Gamma\psi$ and prove $\models_{\mathcal{A}} \Gamma'\psi$. If $\models_{\mathcal{A}} \Gamma\psi$, since $\models_{\mathcal{A}} \Gamma\psi \supset \Gamma''\psi$, we have that $\models_{\mathcal{A}} \Gamma''\psi$ and, since $\models_{\mathcal{A}} \Gamma''\psi \supset \Gamma'\psi$, we have $\models_{\mathcal{A}} \Gamma'\psi$. □

Remark IX.2.1 It is worth noting that the set $[Ax]$ of ground inclusion axiom clauses of a program can be regarded as a *rewriting system* on $\mathcal{C}$, having as rewriting rules the pair (Γ_1, Γ_2) such that $\Gamma_1 \rightarrow \Gamma_2$ belongs to $[Ax]$. In others words, to establish if $\Gamma_1 \overset{*}{\Rightarrow}_{Ax} \Gamma_2$ means to establish if Γ_2 can be derived from Γ_1 by means of a finite number of applications of the rewriting rules of Ax. That is, to establish if Γ_2 belongs to the language $[\Gamma_1]_{Ax} = \{\Gamma \in \mathcal{C}^* : \Gamma_1 \overset{*}{\Rightarrow}_{Ax} \Gamma\}$.

Remark IX.2.2 Given two string Γ_1 and Γ_2, the problem of answering if Γ_2 is a descendant of Γ_1 is known in literature as the *word problem* for the rewriting system. In general the word problem is undecidable since it can be reduced to the Post's Correspondence Problem. Nevertheless, under certain restriction on such systems, it is *decidable.* For example when the system is *complete*, i.e., it is *noetherian* and *confluent* [Book, 1987], or when the language defined by Γ_1 is a *context sensitive language*[4] [Hopcroft and Ullman, 1979].

[3]We denote by $\Gamma_1\Gamma_2$ the concatenation of the modal contexts Γ_1 and Γ_2.

[4]In this case it is shown to be even a PSPACE-complete problem.

These remarks are quite relevant when we have to deal with the implementation of the matching relation in the case when only ground terms may occur within modalities in the program, in the goal and in the axiom clause Ax, and, in particular, no variables may occur within them. In the general case, the problem of implementing the matching relation is more serious, and verifying if a sequence of modalities Γ_1 matches another sequence Γ_2 cannot be simply seen as the problem of determining if Γ_2 can be derived from Γ_1 by applying some rewriting rules. In fact, when the sequences Γ_1 and Γ_2 contain variables, and modalities in the axiom clauses contain variables too, verifying if Γ_1 derives Γ_2 involves some form of *theory unification*.

A goal directed proof procedure

The goal directed proof procedure that we define is *modular* with respect to the underlying inclusion modal logic of a program: the differences among the logics are *factored out* in the derivation relation.

It is worth noting that the proof procedure is an *abstract* one. In particular, we follow [Miller, 1989a], in order to avoid problems with variable renaming and substitutions. Given a program $P = \langle Ds, Ax \rangle$, we denote by $[Ds]$ the set of all ground instances of the set Ds.

Definition IX.2.2 *Let be $\langle Ds, Ax \rangle$ a program in* NemoLOG *and let Γ be an arbitrary modal context. Define $[Ds]$ to be the smallest set satisfying the following conditions:*

- *$Ds \subseteq [Ds]$;*
- *if $\Gamma(\forall x D') \in [Ds]$ then $\Gamma(D'[t/x]) \in [Ds]$ for all ground terms t.*

Hence, given a program $\langle Ds, Ax \rangle$, $[Ds]$ contains ground clauses of the form $\Gamma_b(G \supset \Gamma_h A)$, where Γ_b and Γ_h are arbitrary sequence of modalities (including the empty one), G is a ground goal and A an atomic ground formula.

The *operational derivability* of a *closed goal G* from a *program P* in a *modal context* Γ, is defined by induction on the structure of G. We introduce a proof rule for each kind of goal.

Definition IX.2.3 (Operational Semantics) *Given a program $P = \langle Ds, Ax \rangle$ in* NemoLOG *and a modal context Γ, the operational derivability of a goal G from P in the modal context Γ, written $P, \Gamma \vdash_o G$, is defined by induction on the structure of G as follows:*

1. *$P, \Gamma \vdash_o T$;*
2. *$P, \Gamma \vdash_o A$ if there is a clause $\Gamma_b(G \supset \Gamma_h A) \in [Ds]$ and $\Gamma_b^* \Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, for some Γ_b^* such that $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, and $P, \Gamma_b^* \vdash_o G$;*
3. *$P, \Gamma \vdash_o G_1 \wedge G_2$ if $P, \Gamma \vdash_o G_1$ and $P, \Gamma \vdash_o G_2$;*
4. *$P, \Gamma \vdash_o [t]G$ if $P, \Gamma[t] \vdash_o G$;*

5. $P, \Gamma \vdash_o \exists x G$ *if* $P, \Gamma \vdash_o G[t/x]$*, for some ground term* t*.*

Proving a goal G *from a program* P *amounts to show that* G *is operationally derivable from* P *in the empty modal context* ε*, that is, to show that* $P, \varepsilon \vdash_o G$ *can be derived by making use of the above proof rules.*

While inference rules 1), 3) and 5) are the usual ones for dealing with distinguished symbol T, conjunctive goals and existential goals, rules 2) and 4) are those which deal with modalities. By rule 4), to prove a goal $[t]G$, the modality $[t]$ is added to the current context Γ, and the goal G is proved for the new context $\Gamma[t]$. By rule 2), a clause $\Gamma_b(G \supset \Gamma_h A)$ can be selected from $[Ds]$ to prove an atomic formula A in a given context Γ, if the modalities occurring in front of the clause and in front of the clause head are in a certain relation with Γ, if Γ_b concatenated with Γ_h *derives* Γ according to the properties of modalities specified by the set of axiom clauses Ax.

Example IX.2.1 *(The friends puzzle)* The following is the successful derivation of the first conjunct of the goal $[john][peter]appointment \wedge [peter][john]appointment$ of Example IX.1.2 (the proof of the second conjunct is similar).

1.	$P, \varepsilon \vdash_o [john][peter]appointment$
2.	$P, [john] \vdash_o [peter]appointment$
3.	$P, [john][peter] \vdash_o appointment$
4.	$P, [john][peter] \vdash_o place \wedge time$
5a.	$P, [john][peter] \vdash_o place$
6a.	$P, \varepsilon \vdash_o T$
7a.	success
5b.	$P, [john][peter] \vdash_o time$
6b.	$P, [peter] \vdash_o [peter]time$
7b.	$P, [peter][peter] \vdash_o time$
8b.	$P, \varepsilon \vdash_o T$
9b.	success

We denote with "a" and "b" the two branches which are created by the application of the rule 3) to step 4. Explanation: *1.*: goal; *2.*: by rule 4); *3.*: by rule 4); *4.*: by rule 2), from clause (12) since $[peter][john] \overset{*}{\Rightarrow}_{Ax} [john][peter]$; *5a.*: from 4., by rule 3); *6a.*: by rule 2), from clause (11) since $[peter][john] \overset{*}{\Rightarrow}_{Ax} [john][peter]$; *7a.* by rule 1); *5b.*: from 4., by rule 3); *6b*: by rule 2), from clause (10) since $[wife(peter)] \overset{*}{\Rightarrow}_{Ax} [peter]$ and $[peter][john] \overset{*}{\Rightarrow}_{Ax} [john][peter]$; *7b.*: by rule 4); *8b.*: by rule 2), from clause (9) since $[peter] \overset{*}{\Rightarrow}_{Ax} [peter][peter]$; *9b.*: by rule 1).

Remark IX.2.3 Note that, when the axiom clauses are only of the form $[t_1] \rightarrow [s_1] \dots [s_m]$, that is, there is a single modality on the antecedent, the proof procedure can be simplified. In particular, due to the specificity of the derivation relation, proof rule 2) for atomic formulas can be simplified as follows:

2'. $P, \Gamma \vdash_o A$ if there is a clause $\Gamma_b(G \supset \Gamma_h A) \in [Ds]$ such that, for some Γ_b^* and Γ_h^*, $\Gamma_b^* \Gamma_h^* = \Gamma$, $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, $\Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma_h^*$, and $P, \Gamma_b^* \vdash_o G$;

that is, the current context can be *split* in two parts so that Γ_b derives the first one, and Γ_h derives the second one. This is the kind of semantics it is used in [Baldoni *et al.*, 1993], where a modal logic programming language is proposed to define modularity constructs, and where modalities were ruled by the axioms of $S4$ and K. In the general case, this is not sufficient, and we must require that Γ_b and Γ_h jointly derive the current context Γ. An example is given by the derivation above, where 6*b*. is obtained from 5*b*. and clause (10), by applying rule 2), while it could not be obtained by applying rule 2′).

Example IX.2.2 *(The Fibonacci numbers)* The following is the successful derivation of the goal $[next][next][next]fib(X)$ of Example IX.1.1.

1.	$P, \varepsilon \vdash_o [next][next][next]fib(X)$
2.	$P, [next] \vdash_o [next][next]fib(X)$
3.	$P, [next][next] \vdash_o [next]fib(X)$
4.	$P, [next][next][next] \vdash_o fib(X)$
5.	$P, [next] \vdash_o fib(Y) \wedge [next]fib(Z) \wedge X\ is\ Y + Z$
6a.	$P, [next] \vdash_o fib(Y)$
7a.	success, with $Y = 1$
6b.	$P, [next] \vdash_o [next]fib(Z)$
7b.	$P, [next][next] \vdash_o fib(Z)$
8b.	$P, \varepsilon \vdash_o fib(Y_1) \wedge [next]fib(Z_1) \wedge Z\ is\ Y_1 + Z_1$
9ba.	$P, \varepsilon \vdash_o fib(Y_1)$
10ba.	success, with $Y_1 = 0$
9bb.	$P, \varepsilon \vdash_o [next]fib(Z_1)$
9bb.	$P, \varepsilon \vdash_o [next]fib(Z_1)$
10bb.	$P, [next] \vdash_o fib(Z_1)$
11bb.	success, with $Z_1 = 1$
9bc.	$P, \varepsilon \vdash_o Z\ is\ 0 + 1$
10bc.	success, with $Z = 1$
6c.	$P, [next] \vdash_o X\ is\ 1 + 1$
7c.	success, with $X = 2$

We denote with "a", "b", and "c" the three branches which are created by the application of the rule 3) to step 5. and with "ba", "bb", and "bc" the three branches which are created by the application of the rule 3) to step 8b. Explanation: *1.*: goal; *2.*: by rule 4); *3.*: by rule 4); *4.*: by rule 4); *5.*: by rule 2), from clause (6) since $[always] \stackrel{*}{\Rightarrow}_{Ax} [next]$ and $[next][next][next] \stackrel{*}{\Rightarrow}_{Ax} [next][next][next]$; *6a.*: from 5., by rule 3); *7a.* by rule 1) and 2), from clause (5) since $[next] \stackrel{*}{\Rightarrow}_{Ax} [next]$; *6b.*: from 5., by rule 3); *7b.*: by rule 4); *8b.*: by rule 2), from clause (6) since $[always] \stackrel{*}{\Rightarrow}_{Ax} \varepsilon$ and $[next][next] \stackrel{*}{\Rightarrow}_{Ax} [next][next]$; *9ba.*: from 8b., by rule 3); *10ba.* by rule 1) and 2), from clause (4); *9bb.*: from 8b., by rule 3); *10bb.*: by rule 4); *11bb.* by rule 1) and 2), from clause (5) since $[next] \stackrel{*}{\Rightarrow}_{Ax} [next]$; *9bc.*: from 8b., by rule 3) since $Y_1 = 0$ and $Z_1 = 1$; *6c.*: from 5., by rule 3) since $Y = 1$ and $Z = 1$;

IX.3 Uniform proofs for NemoLOG

In this section we study the relationship between our modal logic programming language and the proof theory of the inclusion modal logics given in Chapter III. In particular, we show that in the case of programs and goals of NemoLOG we can restrict our attention to proofs which are *uniform* as presented in [Miller *et al.*, 1991], where the logical connectives are interpreted as search instructions, so that a uniform proof can be found by a *goal-directed* manner. In order to do this in a easy way, we use the tableau calculus for first-order inclusion modal logic in the form of a cut-free *sequent calculus* but this is only a straightforward syntactic change. As we will observe at the end of the section, the use of prefixed formulae plays an important role which allows us to restrict to uniform proofs (see Remark IX.3.1).

A sequent calculus

We present the *cut-free sequent calculus* for the class of predicative inclusion modal logics. As in the case of tableau method studied in the first part of the thesis, for simplicity, we restrict our attention to a language containing only constant symbols and modal operators labeled with constant symbols. Recall that we denote with $\overline{\mathcal{L}_{FO}}$ the first-order modal language $\mathcal{L}_{FO}$ extended with countably many *new constants* (*parameters*) in order to deal with free variables in the proofs.

Definition IX.3.1 (Sequent calculus) *Let $\mathcal{L}_{FO}$ be a predicative modal language and let $\mathcal{A}$ be a set of inclusion axioms, the sequent calculus for $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$ is shown in Figure IX.1.*

In Figure IX.1, the set $\mathcal{G}$ contains the collection of *accessibility relation formulae* and, intuitively, it is used to keep the accessibility relationships among the worlds represented by means of the prefixes. $R\forall$ and $L\exists$ have the proviso that a_w is a *w-parameter* that does not occur in any formula of the lower sequent. In rule $L\forall$ and $R\exists$ c is any constant of the language $\overline{\mathcal{L}_{FO}}$. The meaning of the rules are simple to understand taking into account the already presented tableau calculus. Note that, in the sequent calculus we do not use signed formulae. Formulae at the left side (the *antecedent*), with respect to the arrow symbol, are the ones interpreted as *true*, while the formulae at the right side (the *consequent*) are the ones interpreted as *false*.

In this sequent calculus there is no need for *structural* rules, since in a sequent $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$ the antecedent and the consequent are sets of statements rather than sequences of statements.

Since T is a distinguished symbol which can be regarded as any propositional tautology, we can assume to have the additional *initial sequent* (*axiom*) $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : T, \Delta$ to deal with this symbol.

A *proof* for the sequent $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$, where Θ and Δ are two set of prefixed signed formulae of $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$, is a *finite tree* constructed using the above rules, having the *root* labeled with $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$ and the *leaves* labeled with initial sequents, i.e. sequents of the form

$$\frac{}{\Theta, w:\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\varphi, \Delta}$$

$$\frac{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\varphi, \Delta}{\Theta, w:\neg\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ L\neg \qquad \frac{\Theta, w:\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\neg\varphi, \Delta}\ R\neg$$

$$\frac{\Theta, w:\varphi, w:\psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}{\Theta, w:\varphi\wedge\psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ L\wedge \qquad \frac{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\varphi, \Delta \quad \Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\psi, \Delta}{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\varphi\wedge\psi, \Delta}\ R\wedge$$

$$\frac{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\varphi, \Delta \quad \Theta, w:\psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}{\Theta, w:\varphi\supset\psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ L\supset \qquad \frac{\Theta, w:\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\psi, \Delta}{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:\varphi\supset\psi, \Delta}\ R\supset$$

$$\frac{\Theta, w:[x/c]\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}{\Theta, w:(\forall x)\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ L\forall \qquad \frac{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:[x/a_w]\varphi, \Delta}{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:(\forall x)\varphi, \Delta}\ R\forall$$

$$\frac{\Theta, w:[x/a_w]\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}{\Theta, w:(\exists x)\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ L\exists \qquad \frac{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:[x/c]\varphi, \Delta}{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:(\exists x)\varphi, \Delta}\ R\exists$$

$$\frac{\Theta, w':\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}{\Theta, w:[t]\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ L[t] \qquad \frac{\Theta \xrightarrow{\mathcal{G}'}_{\mathcal{A}} w':\varphi, \Delta}{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w:[t]\varphi, \Delta}\ R[t]$$

provided that $w\ \rho_t\ w' \in \mathcal{G}$

where w' is *new* on $\mathcal{G}$
and $\mathcal{G}' = \mathcal{G} \cup \{w\ \rho_t\ w'\}$

$$\frac{\Theta \xrightarrow{\mathcal{G}'}_{\mathcal{A}} \Delta}{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ \rho\text{-rule}$$

where $w\ \rho_{s_1}\ w_1,\ \ldots,\ w_{m-1}\ \rho_{s_m}\ w' \in \mathcal{G}$,
$\mathcal{G}' = \mathcal{G} \cup \{w\ \rho_{t_1}\ w'_1,\ \ldots,\ w'_{n-1}\ \rho_{t_n}\ w'\}$,
$w'_1, \ldots, w'_{n-1}$ are *new* on $\mathcal{G}$,
and $[t_1]\ldots[t_n]\varphi \supset [s_1]\ldots[s_m]\varphi \in \mathcal{A}$

Figure IX.1: The sequent calculus for the class of predicative inclusion modal logics.

$\Theta, w : \varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi, \Delta$ or of the form $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : T, \Delta$. We write $\Theta \vdash_{\mathcal{A}} \Delta$ if the sequent $\Theta \xrightarrow{\emptyset}_{\mathcal{A}} \Delta$ has a proof, where Θ and Δ are sets of prefixed signed sentences of $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$ with prefix the initial world i. Furthermore, we say that $\Theta \xrightarrow{\emptyset}_{\mathcal{A}} \Delta$ is *A-valid in a Kripke $\mathcal{A}$-interpretation* $M = \langle W, \mathcal{R}, D, \mathcal{J}, V\rangle$, if, for all $w \in W$, with every constant of the sequent interpreted in $\mathcal{J}(w)$, we have that if $M, w \models^V_{\mathcal{A}} \varphi$, for each $i : \varphi \in \Theta$, then $M, w \models^V_{\mathcal{A}} \psi$, for some $i : \psi \in \Delta$. A sequent $\Theta \xrightarrow{\emptyset}_{\mathcal{A}} \Delta$ is *$\mathcal{A}$-valid* if it is $\mathcal{A}$-valid in each interpretation M of $\mathcal{M}^{\mathcal{A}}_{\mathcal{L}}$.

The sequent calculus above is sound and complete with respect to the Kripke semantics defined in Section V.2.

Theorem IX.3.1 (Soundness and Completeness) *A sequent $\Theta \xrightarrow{\emptyset}_{\mathcal{A}} \Delta$ (with Θ and Δ sets of prefixed signed sentences of $\mathcal{I}^{\mathcal{A}}_{\mathcal{L}_{FO}}$ with prefix i) is valid iff $\Theta \xrightarrow{\emptyset}_{\mathcal{A}} \Delta$ has a proof in the sequent calculus.*

Proof. By Theorem III.3.1 and Theorem III.3.2. □

Uniform proofs

In this section we show that we can restrict our attention to uniform proofs when we consider sequent of the form $i : Ds \xrightarrow{\emptyset}_{\mathcal{A}} i : G$, where $\langle Ds, Ax\rangle$ is a program and G is a goal of our modal logic programming language NemoLOG.

First of all, we can observe that the language NemoLOG does *not allow* existentially quantified clauses *nor* universally quantified goals. Moreover, negation *never* occurs in programs *nor* in goals and implication *never* occurs in goals. For this reason, we can prove the following lemma.

Lemma IX.3.1 *Let Ξ be a proof of a sequent $i : Ds \xrightarrow{\emptyset}_{\mathcal{A}} i : G$ where $\langle Ds, Ax\rangle$ is a program and G a goal of* NemoLOG*. Then Ξ contains no application of the rules $L\neg$, $R\neg$, $R\supset$, $L\exists$ and $R\forall$, where $\mathcal{A} = \{\Gamma\varphi \supset \Gamma'\varphi \mid \Gamma \to \Gamma' \in Ax\}$.*

Proof. Our sequent calculus is *cut-free*. Hence, by the subformula property, derivations are formed entirely from the subformulae of their end sequent. In particular, no negation occurs in Ds and G, and therefore, no application of $R\neg$ or $L\neg$ is allowed in the proof of $i : Ds \xrightarrow{\emptyset}_{\mathcal{A}} i : G$. The same for the implication. Moreover, rules $L\exists$ and $R\forall$ are not applicable too, since in a proof of $i : Ds \xrightarrow{\emptyset}_{\mathcal{A}} i : G$ existentially quantified goals can never occur in the left hand side of a sequent and universally quantified clauses can never occur in the right hand side of a sequent. □

A second observation is about $L\supset$ rule. We show that if we have to prove the sequent $i : Ds \xrightarrow{\emptyset}_{\mathcal{A}} i : G$ then, we can use a weaker version of $L\supset$, namely $L\supset'$, instead of $L\supset$.

Proposition IX.3.1 *Let* Ξ *be a proof of a sequent* $i : Ds \overset{\emptyset}{\rightarrow}_{\mathcal{A}} i : G$ *where* $\langle Ds, Ax\rangle$ *is a program and* G *a goal of* NemoLOG*, then there is a proof* Ξ' *which uses the rule*

$$\frac{\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi \quad \Theta, w : \psi \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta}{\Theta, w : \varphi \supset \psi \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta}\ L{\supset}'$$

instead of $L\supset$ *.*

Proof. We prove the lemma that for all sequent $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta$ in Ξ the following properties hold:

1. there exists a proof of $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta$ which uses the rule $L{\supset}'$ instead of $L\supset$;
2. if Δ has the form $w : \varphi, \Delta'$ (i.e. the sequent has the form $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi, \Delta'$) then there is a proof for $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi$ or for $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta'$ which makes use of $L{\supset}'$ instead of $L\supset$.

In particular, since $i : Ds \overset{\emptyset}{\rightarrow}_{\mathcal{A}} G$ is a sequent which belongs to Ξ the thesis holds. We prove the properties above by induction on height of the proof Υ of $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta$. If the height h of Υ is 1 then Υ is an axiom.

1. Trivial.
2. If $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta$ is $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi, \Delta'$ and it is an axiom then there is a formula $w' : \psi \in \Theta \cap (\{w : \varphi\} \cup \Delta')$ and, in particular, $w' : \psi \in (\{w : \varphi\} \cup \Delta')$. Thus, there are two cases. If $\psi = \varphi$ and $w = w'$ then, $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi$ is provable, while if $\psi \in \Delta'$ then, $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta'$ is provable.

The height of Υ is $h + 1$. By inductive hypothesis the thesis holds for the sequents whose proof has height less or equal to h. We consider the following cases, one for each inference figure in which Υ can terminate.

$R\wedge$, $L\wedge$: Assume that the root inference figure in Υ is $R\wedge$. Hence, Υ is of the form

$$\frac{\overset{\Upsilon_1}{\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi, \Delta'} \quad \overset{\Upsilon_2}{\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \psi, \Delta'}}{\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi \wedge \psi, \Delta'}\ R\wedge$$

1. Trivial, by application of the inductive hypothesis.
2. By inductive hypothesis we have a proof for $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi$ or $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta'$ and a proof for $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \psi$ or $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta'$, that is a proof for $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi$ and $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \psi$ (and hence for $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} w : \varphi \wedge \psi$ by applying $R\wedge$), or $\Theta \overset{\mathcal{G}}{\rightarrow}_{\mathcal{A}} \Delta'$.

The case when the last inference figure is $L\wedge$ is similar.

$R[t]$: Assume that the root inference figure in Υ is $R[t]$. Hence, Υ is of the form

$$\dfrac{\begin{array}{c}\Upsilon_1\\ \Theta \xrightarrow{\mathcal{G}'}_{\mathcal{A}} w' : \varphi, \Delta'\end{array}}{\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : [t]\varphi, \Delta'}\ R[t]$$

1. Trivial, by application of the inductive hypothesis.
2. If we have a proof for $\Theta \xrightarrow{\mathcal{G}'}_{\mathcal{A}} w' : \varphi, \Delta'$ then, we have a proof for $\Theta \xrightarrow{\mathcal{G}'}_{\mathcal{A}} w' : \varphi$ and, by applying the rule$R[t]$, we have a proof for $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : [t]A$.

$L\supset$: Assume that the root inference figure in Υ is $L\supset$. Hence, Υ is of the form

$$\dfrac{\begin{array}{c}\Upsilon_1\\ \Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi, \Delta\end{array} \quad \begin{array}{c}\Upsilon_2\\ \Theta, w : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta\end{array}}{\Theta, w : \varphi \supset \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ L\supset$$

1. Since Υ_1 is shorter than Υ, by inductive hypothesis there is a proof which uses $L\supset'$ instead of $L\supset$ for $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi$ or $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$. Moreover, there is a proof Υ_2' for $\Theta, : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$.
   (a) If there is a proof Υ_1'' for $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi$, which uses $L\supset'$ instead of $L\supset$, we get the following proof for the root sequent

$$\dfrac{\begin{array}{c}\Upsilon_1''\\ \Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi\end{array} \quad \begin{array}{c}\Upsilon_2'\\ \Theta, w : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta\end{array}}{\Theta, w : \varphi \supset \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta}\ L\supset'$$

   (b) If there is a proof for $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$ which uses $L\supset'$ instead of $L\supset$, then, by *weakening*[5] there is a proof for $\Theta, w : \varphi \supset \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$.
2. Assume that $\Theta, w : \varphi \supset \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$ is $\Theta, w : \varphi \supset \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w' : \eta, \Delta'$. Now, we have just proved that

$$\dfrac{\begin{array}{c}\Upsilon_1''\\ \Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi\end{array} \quad \begin{array}{c}\Upsilon_2'\\ \Theta, w : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w' : \eta, \Delta'\end{array}}{\Theta, w : \varphi \supset B \xrightarrow{\mathcal{G}}_{\mathcal{A}} w' : \eta, \Delta'}\ L\supset'$$

   Since, by inductive hypothesis, we have a proof which use $L\supset'$ instead of $L\supset$ for $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi$ and for $\Theta, w : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w' : \eta$ or $\Theta, B \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta'$, we have a proof which use $L\supset'$ instead of $L\supset$ for $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi$ and $\Theta, w : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w' : \eta$ or for $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \varphi$ and $\Theta, w : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta'$. By applying $L\supset'$, we have a proof for $\Theta, w : \varphi \supset \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w' : \eta$ or $\Theta, w : \varphi \supset \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta'$, respectively.

[5]It is easy to show that if $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$ is a provable sequent then, $\Theta, Z \xrightarrow{\mathcal{G}}_{\mathcal{A}} \Delta$, where Z is an arbitrary prefixed formula, is a provable sequent too.

$$\begin{tikzcd}
 & & Ds \models_{\mathcal{A}} G \arrow[dl, shift right, "\begin{array}{l}\textit{Theorem III.3.1}\\ \textit{Theorem III.3.2}\end{array}"'] \arrow[ddrr, shift left, "\textit{see Chapter XI}"] & & \\
 & Ds \vdash_{\mathcal{A}} G \arrow[ur, shift right] \arrow[dl, shift right, "\textit{Theorem IX.3.2}"'] & & & \\
Ds \vdash^{u}_{\mathcal{A}} G \arrow[ur, shift right] & & & & \langle Ds, Ax\rangle, \varepsilon \vdash_{o} G \arrow[uull, shift left]
\end{tikzcd}$$

Figure IX.2: A partial schema of the results about NemoLOG.

$L[t], R\supset\ , L\forall, R\exists, \rho$: Trivial, by application of the inductive hypothesis.

□

From now on we will refer to the sequent calculus with rules $L\wedge$, $R\wedge$, $L[t]$, $R[t]$, $L\supset'$, $R\supset$, $L\forall$, and $R\exists$ and ρ. As a corollary of Proposition IX.3.1 we have the following.

Corollary IX.3.1 *Let Ξ be a proof of a sequent $i : Ds \xrightarrow{\emptyset}_{\mathcal{A}} i : G$, where $\langle Ds, Ax\rangle$ is a program and G a goal of* NemoLOG*. Then, each sequent occurrence in Ξ has a singleton set as its consequent.*

Finally, we show that when we deal with *programs* and *goals* of NemoLOG we can restrict our attention on only *uniform* sequent proofs, if we refer to the notion of uniform proof as presented in [Miller *et al.*, 1991]. This notion provides a natural interpretation of logical connectives as *search* operators in the space of the proofs.

Definition IX.3.2 ([Miller *et al.*, 1991]) *A* uniform proof *is a proof in which each sequent occurrence has a singleton set for its consequent and each occurrence of a sequent whose consequent contains a non-atomic formula is the lower sequent of the inference figure that introduces its top-level connective.*

In our case, we write $\Theta \vdash^{u}_{\mathcal{A}} \Delta$ if $\Theta \vdash_{\mathcal{A}} \Delta$ and the proof is uniform.

Theorem IX.3.2 *Let $\langle Ds, Ax\rangle$ be a program and G a goal of* NemoLOG *then, $Ds \vdash_{\mathcal{A}} G$ if and only if $Ds \vdash^{u}_{\mathcal{A}} G$, where $\mathcal{A} = \{\Gamma\varphi \supset \Gamma'\varphi \mid \Gamma \to \Gamma' \in Ax\}$.*

Proof. (*If* part) Trivial. (*Only if* part) We prove that for all sequent proof Υ of $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta$ in the proof Ξ of $i : Ds \xrightarrow{\emptyset}_{\mathcal{A}} i : G$ there exists a uniform proof Υ' of $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta$. By induction of the height h of the proof of $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta$. If h is 1 then Υ must be an axiom and the thesis holds trivially. The height of Υ is $h + 1$. By inductive hypothesis the thesis holds for proofs with height less of equal to h. We consider the following cases, one for each inference figure in which Υ can terminate.

$L[t], L\wedge, L\forall$: Assume that the root inference figure of Υ if $L[t]$. Hence, Υ is of the form

$$\dfrac{\begin{array}{c}\Upsilon_1\\ \Theta, w'' : \varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta\end{array}}{\Theta, w' : [t]\varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta}\ L[t]$$

By inductive hypothesis there is a uniform proof Υ_1' with root inference figure $\Theta, w'' : \varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta$. Now, we can recognize in Υ_1' all the points where a rule is applied to $w'' : \varphi$. Then, let us change Υ_1' in the following way. Let us assume that Φ is the sub-proof of Υ_1' associated with one of this point with the root inference figure $\Theta', w'' : \varphi \xrightarrow{\mathcal{G}'}_{\mathcal{A}} v : A$. Note that the right end of this sequent must contains an atomic formula. Thus, we add the following step

$$\dfrac{\begin{array}{c}\Phi\\ \Theta', w'' : \varphi \xrightarrow{\mathcal{G}'}_{\mathcal{A}} v : A\end{array}}{\Theta, w' : [t]\varphi \xrightarrow{\mathcal{G}'}_{\mathcal{A}} v : A}\ L[t]$$

obtaining another uniform proof. Now, we can change Υ_1' substituting Φ with the above proof and replacing all formulae $w'' : \varphi$ with $w' : [t]\varphi$ along the path between the sequent $\Theta', w'' : \varphi \xrightarrow{\mathcal{G}'}_{\mathcal{A}} v : A$ and $\Theta, w'' : \varphi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta$ in the proof Υ_1'. Now, we repeat this for all above recognized points.

The case when the last inference figure in Υ are $L\wedge$ and $L\forall$ are similar.

$L\supset'$: Assume that the root inference figure of Υ if $L\supset'$. Hence, Υ is of the form

$$\dfrac{\begin{array}{c}\Upsilon_1\\ \Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w' : \varphi\end{array} \quad \begin{array}{c}\Upsilon_2\\ \Theta, w' : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta\end{array}}{\Theta, w' : \varphi \supset \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta}\ L\supset'$$

By inductive hypothesis there are a uniform proof Υ_1' with root inference figure $\Theta \xrightarrow{\mathcal{G}}_{\mathcal{A}} w' : \varphi$ and a uniform proof Υ_2' with root inference figure $\Theta, w' : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta$. Now, we can recognize in Υ_2' all the points where a rule is applied to $w' : \psi$. Then, let us change Υ_2' in the following way. Let us assume that Φ is the sub-proof of Υ_2' associated with one of this point with the root inference figure $\Theta', w' : \psi \xrightarrow{\mathcal{G}'}_{\mathcal{A}} v : A$. Note that the right end of this sequent must contains an atomic formula. Thus, we add the following step

$$\dfrac{\begin{array}{c}\Phi\\ \Theta', w' : \psi \xrightarrow{\mathcal{G}'}_{\mathcal{A}} v : A\end{array} \quad \begin{array}{c}\Upsilon_2\\ \Theta, w' : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta\end{array}}{\Theta, w' : \varphi \supset \psi \xrightarrow{\mathcal{G}'}_{\mathcal{A}} v : A}\ L\supset'$$

obtaining another uniform proof. Now, we can change Υ_2' substituting Φ with the above proof and replacing all formulae $w' : \psi$ with $w' : \varphi \supset \psi$ along the path between the sequent $\Theta', w' : \psi \xrightarrow{\mathcal{G}'}_{\mathcal{A}} v : A$ and $\Theta, w' : \psi \xrightarrow{\mathcal{G}}_{\mathcal{A}} w : \eta$ in the proof Υ_1'. Now, we repeat this for all above recognized points.

$R\wedge$, $R\exists, R[t], \rho$: Obvious by inductive hypothesis.

Finally, since $i : Ds \xrightarrow{\emptyset}_{\mathcal{A}} i : G$ belongs to Ξ the thesis also holds for it. $\Box$

Remark IX.3.1 Note that the above theorem could be proven only because we make use of a *prefixed* sequent calculus.

In more standard sequent and tableau calculus for modal logics, such as the ones presented in [Fitting, 1983, Chapter 2] and in [Wallen, 1990, Chapter 3], the modal rule $R[t]$ has the effect of deleting some formulae of the "denominator" of the rule to obtain the "numerator" (*destructive* sequent (tableau) systems [Fitting, 1996]). The choice of the formulae is based on both the syntactic structure of the formulae themselves and the properties of the considered logic. Therefore, we can influence the content of a sequent by changing the order of rule application, restricting (or enlarging) the set of formulae available to complete, eventually, the proof (see, for more details, [Wallen, 1990, Chapter 4]). On the contrary, in a prefixed sequent (tableau) calculus this not happened.

In the case of uniform proofs, as shown in [Baldoni *et al.*, 1997a], the problem is that the modal operators in a proof have the effect of *changing* the "context" and, then, they *cannot* be given an interpretation as search operators in the space of proofs (i.e. they do not have a goal directed interpretation) because before using $R[t]$ some applications of left rules may be needed, which is not possible in a uniform proof. In fact, each occurrence of a sequent $\Theta \rightarrow G$ in an uniform proof, where G is not an atomic formula, is obtained by applying the right rule for the main logical connective of G. Instead, in this section, we have shown that a calculus based on prefixed formulae can avoid the necessity of applying left rules before the right rule $R[t]$.

Figure IX.2 summarizes the results of this section. This schema will be completed in Chapter XI, where the soundness and completeness of *operational semantics* with respect to possible-worlds semantics will be proved by means of a *fixed point semantics.*

IX.4 Translating NemoLOG programs into Horn clause logic

NemoLOG has a goal directed operational semantics which has been proved to be sound and complete with respect to the Kripke semantics. The operational derivability of a goal is defined with respect to a notion of *modal context*, which consists of a sequence of modal operators. The modal context keeps track of the new clauses which are added to the program when evaluating implication goals.

The goal directed procedure gives a precise definition of the operation behaviour of a program, and provides a means for executing a program. However the actual implementation of the procedure can raise several problems. The simplest solution of building an interpreter (say in Prolog), may turn out to be inefficient, since the interpreter will have to deal with the modal context.

In this section we present a different approach, based on translating our language into Horn clause logic, so that the translated programs can be executed by any Prolog interpreter or compiler, with the advantage that many features, such as unification or variable renaming, are directly provided. Furthermore, a real program usually needs to use built-in predicates and extra logical features, which, again, are provided by the Prolog environment (as, for instance, *cut*).

The translation methods is based on the idea of implementing directly the operational semantics making explicit reference to the modal context. This is achieved by adding to all predicates an extra argument representing the modal context where the predicate must hold. In particular, a modal context allows us to record the ordering between modalities found in front of goals, during a computation. Note that the notion of modal context plays a role similar to that of *prefixes* of formulas in the tableau method presented in Chapter III. Intuitively, a prefix is a name for a possible world, and the same is for a modal context. A modal context allows us to recognize syntactically whether the worlds being named are accessible or not.

As we will see this approach is closely related to functional translation methods for modal logics [Ohlbach, 1993b] and it is adapted from the translation method for Horn clause languages extended with embedded implication presented in [Baldoni *et al.*, 1996b]. For sake of simplicity, we will be concerned with the case in which the modal operators are only labeled with constant symbols and not with terms[6]. In Appendix A you can find a collection of translated NemoLOG programs taken among the ones presented in this chapter and the following ones.

The translation method

Since universal modal operators are distributive with respect to the conjunction of clauses and goals and due to the converse of Barcan formula that holds, we can assume without loss of generality that a NemoLOG program can always contain universally quantified modalized defined clauses of the following form:[7]

$$\Gamma_b(\Gamma_h A_0 :\!-\ \Gamma_{g_1} A_1, \ldots, \Gamma_{g_m} A_m) \tag{IX.1}$$

and modalized goals of the form:

$$\Gamma_{g_1} A_1, \ldots, \Gamma_{g_m} A_m \tag{IX.2}$$

where $A_1, \ldots, A_m$ are atomic predicates and $\Gamma_b, \Gamma_h, \Gamma_{g_1}, \ldots, \Gamma_{g_m}$ arbitrary sequences of modalities.

Thus, by combining rules 2), 3) and 4) of Definition IX.2.3, we can define the operational derivability of the atomic formulae by means of the new following rule:

[6]Nevertheless, in Appendix A, we have reported some examples which make use of terms with variables.

[7]For readability, we use the standard Prolog syntax extended with modal operators.

$2''$. $P, \Gamma \vdash_o A$
if there is a clause $\Gamma_b(\Gamma_h A :- \Gamma_{g_1} A_1, \ldots, \Gamma_{g_m} A_m) \in [Ds]$ and $\Gamma_b^* \Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, for some Γ_b^*, and $P, \Gamma_b^* \Gamma_{g_1} \vdash_o A_1, \ldots, P, \Gamma_b^* \Gamma_{g_m} \vdash_o A_m$.

The idea for eliminating modalities is based on the structure of rule $2''$) and it is obtained adding to all atomic predicates an argument which represents the modal context where the predicates have to be proved. In others worlds, to move the modal context of operational semantics directly into the predicates.

Let P be a program in NemoLOG and let $derive(\Gamma_b, \Gamma_h, X, Y)$ be a predicate such that it has success if the joint sequence of modalities Γ_b and Γ_h derives the current context X, according to the set of inclusion axiom clauses in P and Definition IX.2.1, and it returns the derived sequence of Γ_b^* by Γ_b in Y. So a clause of the form (IX.1) can be translated as

$$A_0(X) :- derive(\Gamma_b, \Gamma_h, X, Y), A_1(Y \bullet \Gamma_{g_1}), \ldots, A_m(Y \bullet \Gamma_{g_m})^8$$

obtaining a Horn clause, and operational derivability will be defined as SLD resolution. In particular, let $\Gamma_g A$ be a subgoal in the body of a clause, we can translate it in

$$A(Y \bullet \Gamma_g)$$

where Y is a variable which is unified with the current context (the name of the world where $\Gamma_g A$ has to be proved) and linked (denoted by "$\bullet$") with Γ_g for proving A. While a query of the form (IX.2) can be translated as

$$A_1(\Gamma_{g_1}), \ldots, A_m(\Gamma_{g_m}).$$

Note that the added argument "X" in the translated clauses will always be ground during the computation. In fact, since we ask to prove a query in the empty initial modal context, we start each resolution with a goal as $A(\Gamma)$, where Γ does not contain variables. Thus, it is not possible to introduce variables into resolvent, so the the derivation relation works correctly.

We can now give the procedure for translating modalized clauses of NemoLOG into first-order logic, by eliminating modal operators.

Definition IX.4.1 (Procedure for translating into Horn clauses logic) *Let P be a program and a goal in* NemoLOG. *Then, the procedure in Figure IX.3 takes as input the pair P and returns as output P^{tr}, the program obtained by translation of P into Horn clauses logic.*

Note that the sequence $\Gamma' \bullet \Gamma''$ is the concatenation of sequences Γ' and Γ''. Moreover, if A is $p(t_1, \ldots, t_s)$, then $A(X)$ and $A(Y \bullet \Gamma_{g_j})$ are $p(X, t_1, \ldots, t_s)$ and $p(Y \bullet \Gamma_{g_j}, t_1, \ldots, t_s)$, respectively. Finally, the predicate $derive/4$ carries out the derivation relation of Definition IX.2.1, and X and Y are variables.

Let us see how the translation works on the programs IX.1 and IX.2.

[8]Together the proviso that X and Y do not belong to the set of variables of clause (IX.1).

```
begin
    $S := P$;
    for each clause $C \equiv \Gamma_b(\Gamma_h A :- \Gamma_{g_1} A_1, \ldots, \Gamma_{g_m} A_m) \in S$ do
        begin
            $C' := A(X) :- derive(\Gamma_b, \Gamma_h, X, Y),$
                $A_1(Y \bullet \Gamma_{g_1}), \ldots, A_m(Y \bullet \Gamma_{g_m})$;
            $S := (S - \{C\}) \cup \{C'\}$
        end;
    $P^{tr} := S$;
end
```

Figure IX.3: Procedure for translating NemoLOG programs into Horn clause logic.

Example IX.4.1 *(The Fibonacci numbers)* Given the Program IX.1 of the Example IX.1.1, after applying the procedure of Definition IX.4.1, we will obtain the following program P^{tr} (we will denote with ε the empty sequence of modalities).

Program IX.3 : Fibonacci numbers translated.

(1) $fib(X, 0) :-$
  $derive(\varepsilon, \varepsilon, X, Y).$

(2) $fib(X, 1) :-$
  $derive(\varepsilon, [next], X, Y).$

(3) $fib(X, A) :-$
  $derive([always], [next][next], X, Y),$
  $fib(Y \bullet \varepsilon, B),$
  $fib(Y \bullet [next], C),$
  $A\ is\ B + C.$

The goal $[next][next][next]fib(A)$, that is translated into $fib([next][next][next], A)$ succeeds from P^{tr} with the following SLD derivation (denoted by the symbols $\vdash_{SLD}$):

1. $P^{tr} \vdash_{SLD} fib([next][next][next], A)$
2. $P^{tr} \vdash_{SLD} derive([always], [next][next], [next][next][next], Y_0),$
   $fib(Y_0 \bullet \varepsilon, B_0), fib(Y_0 \bullet [next], C_0), A\ is\ B + C$

3a. $P^{tr} \vdash_{SLD} derive([always], [next][next], [next][next][next], Y_0)$
4a. success, with $Y_0 = [next]$
3b. $P^{tr} \vdash_{SLD} fib([next] \bullet \varepsilon, B)$
4b. $P^{tr} \vdash_{SLD} derive(\varepsilon, [next], [next], Y_1)$
5b. success, with $Y_1 = \varepsilon$ and $B = 1$
3c. $P^{tr} \vdash_{SLD} fib([next] \bullet [next], C)$
4c. $P^{tr} \vdash_{SLD} derive([always], [next][next], [next][next], Y_2)$
  $fib(Y_1 \bullet \varepsilon, B_1), fib(Y_1 \bullet [next], C_1), C\ is\ B_1 + C_1$
5ca. $P^{tr} \vdash_{SLD} derive([always], [next][next], [next][next], Y_2)$

6ca. success, with $Y_1 = \varepsilon$
5cb. $P^{tr} \vdash_{SLD} fib(\varepsilon \bullet \varepsilon, B_1)$
6cb. $P^{tr} \vdash_{SLD} derive(\varepsilon, \varepsilon, \varepsilon, Y_3)$
7cb. success, with $Y_3 = \varepsilon$ and $B_1 = 0$
5cc. $P^{tr} \vdash_{SLD} fib(\varepsilon \bullet [next], C_1)$
6cc. $P^{tr} \vdash_{SLD} derive(\varepsilon, [next], [next], Y_4)$
7cc. success, with $Y_4 = \varepsilon$ and $C_1 = 1$
5cd. $P^{tr} \vdash_{SLD} C\ is\ 0 + 1$
6cd. success, with $C = 1$
3d. $P^{tr} \vdash_{SLD} A\ is\ 1 + 1$
4d. success, with $A = 2$

Example IX.4.2 *(The Friends puzzle)* Given the Program IX.2 of the Example IX.1.2, after applying the procedure of Definition IX.4.1, we will obtain the following program P^{tr}.

Program IX.4 : Friends puzzle translated.

(1) $time(X)$:–
 $derive(\varepsilon, [peter], X, Y)$.
(2) $time(X)$:–
 $derive([wife(peter)], [john], X, Y), time(Y \bullet [peter])$.
(3) $place(X)$:–
 $derive(\varepsilon, [peter][john], X, Y)$.
(4) $appointment(X)$:–
 $derive([peter][john], \varepsilon, X, Y)$,
 $place(Y \bullet \varepsilon)$,
 $time(Y \bullet \varepsilon)$.

The goal $[john][peter]appointment$, that is translated into $appointment([john][peter])$ succeeds from P^{tr} with the following SLD derivation:

1. $P^{tr} \vdash_{SLD} appointment([john][peter])$
2. $P^{tr} \vdash_{SLD} derive([peter][john], \varepsilon, [john][peter], Y_0)$,
 $place(Y_0 \bullet \varepsilon), time(Y_0 \bullet \varepsilon)$
3a. $P^{tr} \vdash_{SLD} derive([peter][john], \varepsilon, [john][peter], Y_0)$,
4a. success, with $Y_0 = [john][peter]$
3b. $P^{tr} \vdash_{SLD} place([john][peter] \bullet \varepsilon)$
4b. $P^{tr} \vdash_{SLD} derive([peter][john], \varepsilon, [john][peter], Y_1)$,
5b. success, with $Y_1 = [john][peter]$
3c. $P^{tr} \vdash_{SLD} time([john][peter] \bullet \varepsilon)$
4c. $P^{tr} \vdash_{SLD} derive([wife(peter)], [john], [john][peter], Y_2), time(Y_2 \bullet [peter])$
5ca. $P^{tr} \vdash_{SLD} derive([wife(peter)], [john], [john][peter], Y_2)$,
6ca. success, with $Y_2 = [peter]$
5cb. $P^{tr} \vdash_{SLD} time([peter] \bullet [peter])$
6cb. $P^{tr} \vdash_{SLD} derive(\varepsilon, [peter], [peter][peter], Y_3)$,
7cd. success, with $Y_3 = \varepsilon$

Notice that the steps of the derivations closely correspond to the step of the derivations in Example IX.2.2 and IX.2.1.

The correctness of the whole process of translation is given by the following theorem.

Theorem IX.4.1 (Correctness of the Translation) *Let P be a program and G a goal in* NemoLOG*, then*

$$P, \varepsilon \vdash_o G \textit{ iff } P^{tr} \cup derive/4 \vdash_{SLD} G^{tr}$$

where P^{tr} and G^{tr} are the new program after applying procedures of Definition IX.4.1 and the translated goal, respectively, $\vdash_{SLD}$ is standard operational derivability relation for Horn clause logic, and $derive/4$ is defined on the basis of the set of inclusion axiom clauses in P.

Proof. It follows easily by the above argumentation. □

Remark IX.4.1 This technique has been implemented and tested on several examples. Since the performance of the translated program heavily depends on the predicate *derive*/4, special care was devoted to its implementation. Unfortunately, it is not possible to define a predicate *derive*/4 that works for any set of inclusion axiom clauses of a program because in general, as we have already remarked, the derivation relation for the class of unrestricted grammars is undecidable [Hopcroft and Ullman, 1979]. However, this is not for most of interesting cases such as the ones shown in this chapter and in the Chapter X (see Appendix A).

Translation methods for modal logics have been developed by many authors [Ohlbach, 1993b] as an alternative approach to the development of specific theorem proving techniques and tools. In fact, by translating a modal theorem into predicate logic, it is possible to use a standard theorem prover without the need to build a new one.

The translation methods for modal logics are based on the idea of making explicit reference to the worlds by adding to all predicates an argument representing the world where the predicate holds, so that modal operators can be transformed in quantifiers of classical logic. In particular, in the functional approach [Ohlbach, 1991; Auffray and Enjalbert, 1992], accessibility is represented by means of functions: a modal operator $[m]$ is translated into $\forall F_m$, where F_m is a function of *sort* m, and the worlds will always be represented by a composition of functions, such as $F_{m_1} \bullet \ldots \bullet F_{m_n}$. For instance, the following NemoLOG modalized clause

$$[wife(peter)]([john]time :- [peter]time)$$

will be translated into

$$\forall F_{wife(peter)}((\forall G_{john} time(F_{wife(peter)} \bullet G_{john})) :- \\ (\forall H_{peter} time(F_{wife(peter)} \bullet H_{peter})))$$

This translation is correct if the accessibility relation is assumed to be serial and if the domain of interpretations is constant. We can assume that these conditions hold in our case. The above formula can be transformed to clausal form as follows

$$time(F_{wife(peter)} \bullet G_{john}) :- time(F_{wife(peter)} \bullet c_{peter})$$

where c_{peter} is a Skolem constant of sort *peter*. Note that, since the body is negated, all universally quantified variables in the body have to be skolemized.

The properties of the accessibility relation, such as reflexivity or transitivity, can usually be described with equations which can be translated into a theory unification algorithm. In our case, for instance, a variable $F_{wife(peter)}$ can derive any sequence of functions of sort $wife(peter)$ and *peter*, whereas a variable F_{peter} can derive only a function of sort *peter*.

It is easy to see that this approach closely corresponds to our translation. Sequences of functions in the functional approach correspond to sequences of modal operators in our case and the equational unification is performed by predicate *derive*/4 (in our case we do not need full unification, but only matching).

Chapter X

Applications

One of the aims at defining our modal extension of Horn clause logic is to provide *structuring facilities* as a *basic feature*. Modal operators can be used to this purpose. They can be used to define modules, by associating a modality $[t_i]$ with each module, and, in a more general setting, to provide reasoning capabilities in a multiple agent situation, by associating a modality $[t_i]$ with each agent. Furthermore, this language provides some well-known features of *object-oriented* programming, like the possibility of representing dependencies among modules in a *hierarchy*, and the notion of *self* to reason on this hierarchy.

In the following we show these features through some examples. For readability, we use the standard Prolog syntax extended with modal operators.

X.1 Beliefs, knowledge, and actions representation

We have already remarked that multimodal systems are particularly suited to formalize knowledge and belief operators or to reasoning about actions. NemoLOG inherits this ability, Program IX.2 is an example.

Example X.1.1 is a variant of the above mentioned example that introduce a slightly weaker version of the common knowledge operator in [Halpern and Moses, 1992] already used in Example II.3.4. Example X.1.2, instead, uses modal operators to represent actions.

Example X.1.1 *(Epistemic reasoning and common knowledge: The friends puzzle II)* Let us consider the Example II.3.3, it is reasonable to think that the information that "Peter knows that if John knows the place and the time of their appointment, then John knows that he has an appointment", it is, indeed, a common knowledge. We use the modal operator $[fool]$ to represent this kind of information in Program X.1.

Program X.1 : The friends puzzle II.

$[\texttt{fool}] \rightarrow [\texttt{fool}][\texttt{fool}]$
$[\texttt{fool}] \rightarrow \varepsilon$
$[\texttt{fool}] \rightarrow [\texttt{peter}]$
$[\texttt{fool}] \rightarrow [\texttt{john}]$

```
[fool] → [wife(peter)]
[peter][john] → [john][peter]
[peter] → [peter][peter]
[peter] → ε
[john] → [john][john]
[john] → ε
[wife(peter)] → [peter]
[wife(peter)] → [wife(peter)][wife(peter)]
[wife(peter)] → ε

[peter]time.
[peter][john]place.
[wife(peter)]([john]time :- [peter]time).
[fool](appointment :- place, time).
```

Remark X.1.1 As already remarked above, our modal operator $[fool]$ can be taken as a weaker version of the common knowledge operator. In fact, in the possible-worlds semantics associated, differently that the one in [Halpern and Moses, 1992], the accessibility relation associated to $[fool]$ includes the transitive and reflexive closure of the union of the accessibility relations associated with the other epistemic operators and *not to be* equal to it (see also Remark VI.2.1). That means that $[fool]$ cannot be regarded as a common knowledge operator, though it shares some of its properties. In particular, in our example, the formula:

$$\varphi \wedge [fool](\varphi \supset [peter]\varphi \wedge [john]\varphi \wedge [wife(peter)]\varphi) \supset [fool]\varphi$$

(the *induction axiom* for common knowledge) is not valid in the possible-worlds semantics of our language, while it is expected to be a valid formula when $[fool]$ is a common knowledge operator. In [Genesereth and Nilsson, 1987] a similar weaker version of common knowledge operator is suggested. To explain this notion of common knowledge, in [Genesereth and Nilsson, 1987] a *fictitious knower* has been assumed, sometimes called *any fool*. What *any fool* knows is what all other agents know, and all agents know that others know (and so on). In other words, instead of regarding common knowledge as an operator over beliefs of agents, it is regarded as a *new agent* which interacts with the others.

The following example presents a modal version of the well-known "shooting problem". The solution proposed, differently than [Baldoni *et al.*, 1997b], is *monotonic* and the *frame axiom* is explicitly represented in the clauses.

Example X.1.2 *(Reasoning about actions: The shooting problem)* Assume that our language contains a K modality $[a]$ for each possible atomic action a, and modalities $[s_1; s_2]$ to represent sequences of actions and a modality $[\varepsilon]$ to represent the initial state. The set $\mathcal{A}$ will contain the logical axioms $[s_1][s_2]\alpha \supset [s_1; s_2]\alpha$, for all action sequences s_1 and s_2. We formalize the well known "shooting problem" with the Program X.2.

Program X.2 : Shooting problem.

(1) $[\mathtt{S_1}][\mathtt{S_2}] \rightarrow [\mathtt{S_1};\mathtt{S_2}]$

(2) $[\varepsilon]$`alive`.
(3) $[\varepsilon]$`unloaded`.
(4) `[S]([shoot]dead :- loaded)`.
(5) `[S; load]loaded`.
(6) `[S]([A]alive :- alive,` $\mathtt{A} \neq$ `shoot)`.
(7) `[S]([shoot]alive :- alive, unloaded)`.
(8) `[S]([A]loaded :- loaded,` $\mathtt{A} \neq$ `shoot)`.
(9) `[S]([A]unloaded :- unloaded,` $\mathtt{A} \neq$ `load)`.

Clauses (2) and (3) represent the *initial facts*, the clauses (4) and (5) the *causal rules*, and the clauses (6)-(9) the *frame axioms*. In this example it is worth using modalities labeled with terms which contains variables to represent arbitrary sequences of actions. The goal $G = [\varepsilon; load; wait; shoot]dead$ succeeds with the following derivation.

1. $\varepsilon \vdash_o [\varepsilon;\mathtt{load};\mathtt{wait};\mathtt{shoot}]\mathtt{dead}$
2. $[\varepsilon;\mathtt{load};\mathtt{wait};\mathtt{shoot}] \vdash_o \mathtt{dead}$
3. $[\varepsilon;\mathtt{load};\mathtt{wait}] \vdash_o \mathtt{loaded}$ by clause (3) and $\mathtt{S} = \varepsilon;\mathtt{load};\mathtt{wait}$ and
   $[\varepsilon;\mathtt{load};\mathtt{wait}][\mathtt{shoot}] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\varepsilon;\mathtt{load};\mathtt{wait};\mathtt{shoot}]$,
4. $[\varepsilon;\mathtt{load}] \vdash_o \mathtt{loaded}$ by clause (7) and $\mathtt{S} = \varepsilon;\mathtt{load}$, $\mathtt{A} = \mathtt{wait}$ and
   $[\varepsilon;\mathtt{load}][\mathtt{wait}] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\varepsilon;\mathtt{load};\mathtt{wait}]$,
5. success, by clause (4) and $\mathtt{S} = \varepsilon$ and $[\varepsilon;\mathtt{load}] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\varepsilon;\mathtt{load}]$.

It is interesting to note the also the goal $G' = [\mathtt{Z}]\mathtt{dead}$ succeeds with $\mathtt{Z} = \varepsilon;\mathtt{load};\mathtt{shoot}$.

X.2 Defining modules

One of the main motivations in defining this language comes from the need of *structuring facilities* to enhance *modularity*, *readability* and *reusability* of logic programs. This problem has been addressed in the literature using many different approaches (like the *meta-level* approach [Bowen and Kowalski, 1982; Brogi *et al.*, 1992], the *algebraic* approach [O'Keefe, 1985; Mancarella and Pedreschi, 1988; Brogi *et al.*, 1994], and the approach based on use of *higher-order logic* [Nait Abdallah, 1986; Chen, 1987]) and, in particular, it has been tackled by extending the language of Horn clauses with implications embedded in goals, as proposed in [Miller, 1989a; Monteiro and Porto, 1989; Giordano *et al.*, 1992; Lamma *et al.*, 1993; Giordano and Martelli, 1994] (see [Bugliesi *et al.*, 1994] for a survey of the different approaches).

In this section we show, through some examples, that the language we have introduced is well suited to define module constructs. And, in particular, it allows to introduce *structuring constructs* in logic programs while *preserving* their logical semantics.

The key idea is to use a modal operator $[m_i]$ of type K for representing what is true in a module, i.e. each label m_i can be regarded as a module name (see also [Baldoni *et al.*, 1993; Baldoni *et al.*, 1997a]).

Flat collection of modules

As we have mentioned above, a modal operator $[m_i]$ of type K can be associated with a module and can be used to represent what is true in it. In this case, the term m_i can be regarded as a *module name*. This provides a simple way to define a *flat collection of modules* and to specify the proof of a goal in a module. In particular, if Ds is a set (conjunction) of clauses we may define the clauses in Ds as belonging to module m_i through the *module definition*

$$[export][m_i]Ds.$$

The modality $[export]$ of type $KT4$ in front of the module definition is needed to make the definition visible in *any* context (and, in particular, from inside other modules). To this purpose the *inclusion* axiom

$$I(export, m_i) : [export]\varphi \supset [m_i]\varphi$$

is required. To prove a goal G in module m_i, we have simply to write the goal

$$[m_i]G.$$

Initially, we assume that clauses in a module must have the form $G \supset A$, where G may contain occurrences of goals $[a_i]G$.

Example X.2.1 *(Bubblesort I)* Consider the simple Program X.3 containing two module definitions. For readability, we put module name in front of the sequence of clauses of the module, rather than in front of each one.

Program X.3 : Bubblesort I.

```
    [export] → ε
    [export] → [export][export]
    [export] → [list]
    [export] → [sort]

    [export][list] {
        append([ ], X, X).
        append([X|Y], Z, [X|Y1]) :−
            append(Y, Z, Y1). ...} % End of module list.

    [export][sort] {
        busort(L, S) :−
            [list]append(X, [A, B|Y], L),
            B < A,
            [list]append(X, [B, A|Y], M),
            busort(M, S).
        busort(S, S). ...} % End of module sort.
```

The module `list` contains the definition of `append` and other predicates on `list`, while the module `sort` contains the definition of the predicate `busort` for ordering a list according to the bubblesort algorithm.

The goal $[\texttt{sort}]\texttt{busort}([2,1,3],\texttt{S})$ succeeds with answer $\texttt{S} = [1,2,3]$. Note that, in its computation, the subgoal $\texttt{append}(\texttt{X},[\texttt{A},\texttt{B}|\texttt{Y}],[2,1,3])$ has to be proved in the context $[\texttt{sort}][\texttt{list}]$ and, hence, it can only be proved by making use of the clauses in the module `list`. In fact, the clauses in module `sort` cannot be used in the context $[\texttt{sort}][\texttt{list}]$, since all of them are prefixed by the sequence of modalities $[\texttt{export}][\texttt{sort}]$ which does not derive $[\texttt{sort}][\texttt{list}]$ by means of Ax.

Composition of modules: exporting information

In the previous section modules are *closed* environments, and they cannot be *composed.* Thus, in this case, the query $[m_1][m_2]G$ which succeeds if G can be proved from the clauses in module m_2, is completely equivalent to the query $[m_2]G$.

However, our language also enables modules to be defined as *open* environments, so that proving the query $[m_1][m_2]G$ amounts to prove the goal G in the *composition* of modules m_1 and m_2. Languages providing modularity features of this kind have been presented in [Miller, 1989a; Monteiro and Porto, 1989; Lamma *et al.*, 1993]. Also, a similar point of view has been taken in [Bugliesi, 1992], where a declarative characterization of inheritance is defined, and in [McCabe, 1992], where an extension of logic programming is proposed to capture the main features of *object-oriented* programming.

When a module is regarded as being open, it is allowed to *export* some information to the external environment. Consider for instance the query $[m_1][m_2][m_3]G$, the goal G must be proved in the composition of modules m_1, m_2 and m_3. The ordering of modules in the query determines the direction in which information is exported: each module can export information to the modules following it in the sequence.

In our language, different forms of module composition can be obtained by making use of the already introduced modal operator $[export]$ to *control* the information (either clauses or derived facts) that can be exported by a module. In particular, we can make a distinction among: clauses that are *local* to the module in which they are defined,

$$G \supset A$$

(as in the Example X.2.1), clauses that are *wholly exported* by the module,

$$[export](G \supset A)$$

(we call these clauses *dynamic*), and clauses that *only export their head* (consequences),

$$G \supset [export]A$$

(we call these clauses *static*). This feature allows to model different kinds of modules presented in the literature (so that in each situation the kinds of module that suit better can be adopted).

Example X.2.2 *(Bubblesort II)* Let us consider Program X.4 another formulation of the previous example, which makes use of static clauses.

Program X.4 : Bubblesort II.

```
[export] → ε
[export] → [export][export]
[export] → [list]
[export] → [sort]

[export][list] {
    [export]append([ ], X, X).
    [export]append([X|Y], Z, [X|Y1]) :−
        append(Y, Z, Y1). ...} % End of module list.

[export][sort] {
    [export]busort(L, S) :−
        append(X, [A, B|Y], L),
        B < A,
        append(X, [B, A|Y], M),
        busort(M, S).
    [export]busort(S, S). ...} % End of module sort.
```

In this formulation, differently from the previous one, the subgoals `append` in the body of the first clause for `busort` are not preceded by the modal operator [`list`], and hence, they must be proved in the current context, in which a definition of the `append` predicate must be provided. This can be done by asking the query [`lists`][`sort`]`busort`([2, 1, 3], `S`), that is, by asking for a proof of the goal `busort`([2, 1, 3], `S`) in the composition of the two modules `list` and `sort`. The query succeeds from the program.

The predicate `append` is exported from module `list` (which contains static clauses) and thus it is visible from `sort`. Note that the body of the second `append` clause must be proved only in the module `list` and its proof cannot use any predicate defined within module `sort`.

Remark X.2.1 When, as in Example X.2.2, static visibility rules are used, our language has a behavior similar to that of the language proposed in [Monteiro and Porto, 1989; Monteiro and Porto, 1990]. A difference between the two languages is that their language adopts predicate overriding between modules, that is, given a query $[m_1][m_2][m_3]G$, the clause definitions for the predicate p in m_3 override (cancel) the definitions of p in m_2 and in m_1. In our language, on the other hand, the definitions of a predicate may be spread in different modules, and all of them can be used.

Nested modules

In the previous sections we have seen some programs consisting of a *flat* collection of modules. However, NemoLOG also allows *nested* modules to be defined. By exploiting the feature that clauses can be preceded by an arbitrary sequence of modal operators, we can generalize module definitions $[export][m_i]Ds$, given above, as follows:

$$[export][m_i][m_j]Ds$$

where the module m_j is defined *locally* to m_i, and it becomes visible whenever m_i is entered.

The following example (from [Goldberg and Robson, 1983]) shows how we can use nested modules.

Example X.2.3 *(Dictionary)* We define a *dictionary* of pairs (*name, value*) with two possible implementations. The first one, named `fast`, makes use of a *search tree* and can be used for big dictionaries, where fast access is important. The second one, named `small`, makes use of a *list* and can be used for small dictionaries, if we want to minimize the space needed to store information. The formulation is given by Program X.5.

Program X.5 : Dictionary.

```
[export] → ε
[export] → [export][export]
[export] → [dictionary]
[export] → [small]
[export] → [fast]

[export][dictionary] {

    [export](getvalue(Name, Value, Dictionary) :-
        not_empty(Dictionary),
        search(Name, Dictionary, Value)).
    [export](putvalue([Name, Value], Dictionary, NewDictionary) :-
        not_member(Name, Dictionary),
        insert([Name, Value], Dictionary, NewDictionary)).

    [fast] {
        not_empty([[Name, Value], L, R]).
        search(Name, [[Name, Value], _, _], Value).
        search(Name, [[Name1, _], L, R], Value) :-
            Name < Name1,
            search(Name, L, Value). ... } % End of module fast.

    [small] {
        not_empty([[Name, Value]|L]).
        search(Name, [[Name, Value]|_], Value).
        search(Name, [[Name1, _]|L], Value) :-
            Name ≠ Name1,
            search(Name, L, Value). ... } % End of module small.

    ... } % End of module dictionary.
```

The module `dictionary` contains the definition of `getvalue`, which returns the *value* associated with a *name*, and `putvalue`, which insert a new pair (*name*, *value*) in a dictionary if it is not already a member of it. The module `dictionary` also *contains* two nested modules, `fast` and `small`, which describe the predicates used in the definition of `getvalue` and `putvalue`, in the case we wish to use a fast dictionary or a small dictionary, respectively. Then, we can retrieve a *value* associated to a *name* in a fast dictionary by asking the goal

$$[\texttt{dictionary}][\texttt{fast}]\texttt{getvalue}(Name, Value)$$

Note that we can use module `fast` *only* when module `dictionary` is entered. In fact, using module `fast` (respectively `small`) is meaningful only when it is composed with module `dictionary`. Observe, moreover, that the usage of a dynamic clause for predicates `getvalue` and `putvalue` in module `dictionary` is due to the fact that they use predicates defined in module `fast` (respectively, `small`).

Parametric modules

Parametric modules are an important features of a module system. They allow to enhance the modularity [Giordano *et al.*, 1994; Hill, 1993] as well as to support some aspects of object-oriented [McCabe, 1992; Monteiro and Porto, 1990; Lamma *et al.*, 1993]. In NemoLOG a modalized defined clause can be of the form

$$\forall x[t(x)](Ds(x))$$

where the variable x is free in the set of clauses $Ds(x)$. Indeed, the above formula is also the definition of a module and, in particular, of a *parametric* module. In NemoLOG, parametric modules can be obtained by *sharing* some variables between the label of the modalities (the name of a module) and their associated clauses (the body of a module).

Example X.2.4 *(Bubblesort III)* Let us consider the module definition in Program X.6

Program X.6 : Bubblesort III.

```
[export] → ε
[export] → [export][export]
[export] → [list]
[export] → [ascending]
[export] → [descending]
[export] → [sort(ascending)]
[export] → [sort(descending)]

[export][list]{
    append([ ], X, X).
    append([X|Y], Z, [X|Y1]) :−
        append(Y, Z, Y1). ...} % End of module list.
```

```
[export][ascending]{
    ordered(X,Y) :- X < Y. ...} % End of module ascending.

[export][descending]{
    ordered(X,Y) :- X > Y. ...} % End of module descending.

[export][sort(Order)]{
    busort(L,S) :-
        [list]append(X,[A,B|Y],L),
        [Order]ordered(B,A),
        [list]append(X,[B,A|Y],M)
        busort(M,S).
    busort(S,S). ...} % End of module sort.
```

As already seen, the module `lists` contains the definition of `append` and the other predicates on lists, while the module `sort(Order)` contains the definition of the predicate `busort` as in Program X.3. In order to parameterize the algorithm with respect to the *type* of the order, we introduce two modules, named `ascending` and `descending`, which contain two different definition of the predicate `ordered`. Now, we can specify a particular order through the variable `Order`. Thus, the goal

```
[sort(ascending)]busort([2,1,3],S)
```

succeeds with answer `S = [1,2,3]`, while the goal

```
[sort(descending)]busort([2,1,3],S)
```

succeeds with answer `S = [3,2,1]`.

Nested and parametric modules can be used in supporting the notion of an *abstract data-type*. Program X.5 and X.6 are examples of this. The following example shows how to extend the Program X.6 in order to deal with *pairs* of natural number instead of only simple natural number.

Example X.2.5 *(Bubblesort IV)* We can extend Program X.6 to deal also with pairs of number simply adding the module in Program X.7

Program X.7 : Bubblesort IV.

```
[export] → [cartesian(Ord1,Ord2)]

[export][cartesian(Ord1,Ord2)]{
    ordered([X,Y],[U,V]) :-
        [Ord1]ordered(X,U).
    ordered([X,Y],[X,V]) :-
        [Ord2]ordered(Y,V). ...} % End of module cartesian.
```

The module `cartesian(Ord1,Ord2)` specifies the predicate `ordered` for pairs of number. Note that this module is parametric so that we can choose by means of the variables `Ord1` and `Ord2` the kind of ordering for each element of the pairs. The goal

```
[sort(cartesian(ascending,descending))]busort([[3,4],[1,6],[3,2],[10,5]],S)
```

succeeds with answer `S = [[1,6],[3,4],[3,2],[10,5]]`.

X.3 Inheritance and hierarchies

Another important problem related with providing support for software engineering is the integration of logic programming and *object-oriented paradigms* [McCabe, 1992; Bugliesi, 1992] (see also [Bugliesi *et al.*, 1994, Section 3.6] and [Turini, 1995]). A significant proposal to tackle this problem is the *class template language* presented in [McCabe, 1992], where the idea of representing an object as a first-order logic theory is exploited. McCabe interprets *attributes* and *methods* of an object as a set of formulae. Classes are introduced by means of *parametric modules* whose parameters play the role of instance variables of the object-oriented languages. *Class rules* allow to specify the structure of the classes and, thus, the *inheritance hierarchy.*

From a different perspective, in the following examples, we show how modal logics can be used to obtain some features of object-oriented paradigms, although we do not deal with the *state* of objects. In particular, *hierarchical dependencies* among modules can be represented both by means of *nested modules* and by inclusion axiom schemas. For example, if $[m_i]Ds_i$ and $[m_j]Ds_j$ represent two modules, the inclusion axiom

$$[m_i]\varphi \supset [m_j]\varphi$$

says that all the clauses of module m_i are *exportable* into module m_j; in different words m_j is a more specific subclass of m_i. Besides, a behavior similar to the use of *self* can be obtained by means of the previously introduced modal operator [*export*] and using dynamic clauses.

Example X.3.1 *(Animal taxonomy I)* This is an example of the usefulness of dynamic clauses in nested modules. It is taken from [Brogi *et al.*, 1990b] and describes inheritance in a hierarchy of modules. Program X.8 describes a simple taxonomy that has three levels: the root (*animal*), which contains the subclasses *horse*, *bird*, and *tweety*, which is a subclass of *bird.*

Program X.8 : Animal taxonomy I.

```
[export] → ε
[export] → [export][export]
[export] → [animal]
[export] → [bird]
[export] → [tweety]
```

```
[export][animal] {
    [export]mode(walk).
    [export](mode(run) :− no_of_legs(X), X ≥ 2).
    [export](mode(gallop) :− no_of_legs(X), X = 4).

    [horse] {
        [export]no_of_legs(4).
        [export]covering(hair). ...} % End of module horse.

    [bird] {
        [export]no_of_legs(2).
        [export]covering(feather).
        [export]mode(fly).

        [tweety] {
            [export]owner(fred). ...} % End of module tweety.
        ...} % End of module bird.
    ...} % End of module animal.
```

The goal

$$[\texttt{animal}][\texttt{bird}][\texttt{tweety}]\texttt{mode}(\texttt{run})$$

succeeds, since the clause defining `mode(run)` is exported by the module `animal` and its body can be evaluated in the current context, including module `bird` which contains the information `no_of_legs(2)`. The goal would fail, if the modality [*export*] in front of the clause

$$[\texttt{export}](\texttt{mode}(\texttt{run}) :- \texttt{no\_of\_legs}(\texttt{X}), \texttt{X} \geq 2)$$

in module `animal` were omitted. By using clauses preceded by the operator [*export*] (dynamic clauses) we can achieve a result somewhat similar to the use of *self* in object-oriented languages, by allowing methods of a class to use information coming from a more specific class.

In the following example we show how to obtain the same description of Example X.3.1 but using *inclusion axioms* to describe the hierarchical dependency among modules instead of nested modules.

Example X.3.2 *(Animal Taxonomy II)* Let us consider again the four classes *animal*, *horse*, *bird* and *tweety*. Since what is true for animals is also true for birds and horses, the *bird* and *horse* class inherit from the *animal* class. Moreover, the class *tweety* inherits from *bird* and thus from *animal*. To model this situation, we use the following set of *inclusion axioms* for defining the inheritance rules:

$$\begin{array}{ll} I(animal, horse): & [animal]\alpha \supset [horse]\alpha \\ I(animal, bird): & [animal]\alpha \supset [bird]\alpha \\ I(bird, tweety): & [bird]\alpha \supset [tweety]\alpha \end{array}$$

Thus, the Program X.8 becomes the following.

Program X.9 : Animal taxonomy II.

```
[animal] → [horse]
[animal] → [bird]
[bird] → [tweety]

[animal]{
    mode(walk).
    mode(run) :− no_of_legs(X), X ≥ 2.
    mode(gallop) :− no_of_legs(X), X = 4. ...} % End of module animal.

[horse]{
    no_of_legs(4).
    covering(hair). ...} % End of module horse.

[bird]{
    no_of_legs(2).
    covering(feather).
    mode(fly). ...} % End of module bird.

[tweety]{
    owner(fred). ...} % End of module tweety.
```

The goal [tweety]mode(run) succeeds, since the clause defining mode(run) is inherited by the class *tweety* from *animal*.

Note that, Program X.9 enjoys some distinctive characteristics with respect to the Program X.8:

- we do not need to use dynamic clause inside a module for export its clauses; and
- we do not need to specify the whole hierarchy to query something about a class (*tweety* in the example) even with statically configured module systems [Brogi *et al.*, 1990b]. Inclusion axiom clauses works like *class rules* in class template language of McCabe.

Finally, it is also interesting to note that we can ask goals like [X]mode(fly). In fact, it succeeds with answers X = bird and X = tweety.

The following example, inspired from [McCabe, 1992], shows another interesting feature of NemoLOG related to the use of parametric modules and axiom clauses.

Example X.3.3 The class *human*(S, A) is a subclass of *animal*. It is defined by a parametric module whose parameters allow to specify the attribute *age* and *sex* of a particular *instance* of a human. Furthermore, we define the class *mathematician* that is not subclass of any other class.

Program X.10 : Humans.

```
[animal] → [human(S, A)]

[human(S, A)]{
    sex(S).
    age(A).
    no_of_legs(2).
    likes(logic) :−
        sex(male),
        age(Ag),
        Ag < 40.
    likes(logic) :−
        sex(female).
    ...} % End of module human.

[mathematician]{
    likes(logic).
    likes(math).
    ...} % End of module mathematician.

[human(male, 30)] → [peter]

[human(female, 42)] → [jane]

[human(male, 45)] → [john]
[mathematician] → [john]
```

Now, the axiom clauses are used both to specify a hierarchical structure among modules and to create particular instance of the class *human*. Then, `peter`, `jane`, and `john` are the instance of the class *human* and inherits all its content. Thus, the goal

$$[\texttt{peter}](\texttt{mode(walk)} \wedge \texttt{likes(logic)})$$

succeeds because `peter` is a *human* aged 30 and because he is an *animal* and, then, he can walk. It is interesting to note that, despite the fact that `john` is a *human* aged 42, the goal

$$[\texttt{john}](\texttt{mode(walk)} \wedge \texttt{likes(logic)})$$

succeeds. In fact, `john` is also a *mathematician* and, then, inherits both from the class *human* and from the class *mathematicial* (*multiple inheritance*).

Evolving and conservative systems with dynamic or static configuration of modules

In [Brogi *et al.*, 1990a; Brogi *et al.*, 1990b; Lamma *et al.*, 1993] a general unifying framework for structuring logic programs, called *Ctx_Prolog*, is presented. It is inspired by the works in [Monteiro and Porto, 1989; Miller, 1989a] and it is aimed at giving a framework in which different proposals for structuring logic programs can be described and compared.

A program in Ctx_Prolog is a collection of named modules (*unit*) while goals are proved in variable sets of clauses (*context*) obtained by suitably combining units by means of the *extension operators* ">>" (*cactus extension*) and ">>>" (*linear extension*). In particular, in Ctx_Prolog a distinction is made between *statically* and *dynamically* configured systems and between *conservative* (or *nested*) and *evolving* (or *global*) *policies* to establish bindings of predicate calls (this distinction roughly corresponds to the distinction between *static* and *dynamic* visibility rules for non-local predicate definitions in [Giordano and Martelli, 1992; Giordano and Martelli, 1994]).

A statically configured system is defined as a system where hierarchies among units are specified when units are defined. In these systems the context in which a unit is used does not depend on the dynamic sequence of goals but is always fixed when the unit is defined. For instance, in [Lamma *et al.*, 1993], to specify that whenever a unit m_1 is used, it is used only in the context of the modules m_2, m_3, and m_4 a definition of the form

$$\texttt{unit(m1,closed(}[\texttt{m}_2,\texttt{m}_3,\texttt{m}_4]))$$

takes place in the program. On the contrary, the context of unit m_1 can be different in different queries.

In our language, nested modules allows to describe a sort of statically configured modules. Let us consider the Example X.3.1, the module `tweety` is visible only if `bird` and `animal` are entered. However, nested modules does not model the meaning of static configuration of modules as given in [Lamma *et al.*, 1993], In fact, since `animal` is exportable, the sequence of modules [`animal`][`bird`][`tweety`] can be the suffix of different contexts and, thus, `tweety` can inherit information not only from `animal` and `bird`. On the other hand, module `animal` needs to be defined exportable in order to make it visible inside other modules.

Nevertheless, statically configured modules can be allowed by introducing a new modal operator [*public*] of type $S4$ to control the information that can be exported by a module (instead of [*export*]) and the modal operator [*closed*] of type K to make a context closed.

Example X.3.4 *(Statically and dynamically configured systems)* Let us consider three modules, named respectively m_1, m_2 and m_4. Modules m_1 and m_2 are static, while module m_3 is dynamic [Brogi *et al.*, 1990a, Example 6].

Program X.11 : Statically and dynamically configured systems.

(1)	$[\texttt{export}] \rightarrow \varepsilon$	(2)	$[\texttt{public}] \rightarrow \varepsilon$
(3)	$[\texttt{export}] \rightarrow [\texttt{export}][\texttt{export}]$	(4)	$[\texttt{public}] \rightarrow [\texttt{public}][\texttt{public}]$
(5)	$[\texttt{export}] \rightarrow [\texttt{closed}]$		
(6)	$[\texttt{export}] \rightarrow [\texttt{m}_1]$	(7)	$[\texttt{public}] \rightarrow [\texttt{m}_1]$
(8)	$[\texttt{export}] \rightarrow [\texttt{m}_2]$	(9)	$[\texttt{public}] \rightarrow [\texttt{m}_2]$
(10)	$[\texttt{export}] \rightarrow [\texttt{m}_3]$	(11)	$[\texttt{public}] \rightarrow [\texttt{m}_3]$

(12) $[\texttt{m}_1][\texttt{public}] \rightarrow [\texttt{m}_2]$

```
        [export][m1] {
(13)        [public]b. }
        [export][m2] {
(14)        [public]a :− b.}
(15)        [public]a′ :− b′.}

        [export][m3] {
(16)        [public]c :− [closed][m2]a.
(17)        [public]c′ :− [closed][m2]a′.
(18)        [public]b′. }
```

In this way, $\mathtt{m}_2$ always inherits the "*public*" information of module $\mathtt{m}_1$ (by means of the inclusion axiom clause $[\mathtt{m}_1][\mathtt{public}] \rightarrow [\mathtt{m}_2]$) but not the other ones. The goal $[\mathtt{m}_3]\mathtt{c}$ has the following successful derivation:

1. $\varepsilon \vdash_o [\mathtt{m}_3]\mathtt{c}$
2. $[\mathtt{m}_3] \vdash_o \mathtt{c}$
3. $[\mathtt{m}_3] \vdash_o [\mathtt{closed}][\mathtt{m}_2]\mathtt{a}$ by clause (16) and $[\mathtt{export}][\mathtt{m}_3] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\mathtt{m}_3]$, $[\mathtt{m}_3][\mathtt{public}] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\mathtt{m}_3]$
4. $[\mathtt{m}_3][\mathtt{closed}] \vdash_o [\mathtt{m}_2]\mathtt{a}$
5. $[\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_2] \vdash_o \mathtt{a}$
6. $[\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_2] \vdash_o \mathtt{b}$ by clause (14) and $[\mathtt{export}][\mathtt{m}_2] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_2]$,
   $[\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_2][\mathtt{public}] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_2]$
7. success, by clause (13) and $[\mathtt{export}][\mathtt{m}_1] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_1]$,
   $[\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_1][\mathtt{public}] \overset{*}{\Rightarrow}_{\mathtt{Ax}} [\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_2]$

On the other hand, the goal $[\mathtt{m}_3]\mathtt{c}'$ does not succeed since clause (18) is not visible inside the context $[\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_2]$ because $[\mathtt{export}][\mathtt{m}_3][\mathtt{public}]$ does not derive $[\mathtt{m}_3][\mathtt{closed}][\mathtt{m}_2]$. In other worlds, the modal operator [*closed*] in front of a goal has the effect of closing a context.

We said that the proposal in [Brogi *et al.*, 1990a; Brogi *et al.*, 1990b; Lamma *et al.*, 1993] makes a a distinction between conservative and evolving policies. More precisely, it is possible in Ctx_Prolog to put the symbol "#" in front of the atomic goals. $\#p$ means that p is a *lazy* atom and, operationally, it has to be solved dynamically from the current context of modules. This gives the evolving policy. On the other hand, if the operator "#" is not used in front of an atomic goal (*eager* atom), it means that the atom coming from a module has to be solved statically only using clauses defined in that module or in externally nested modules. This gives the conservative policy.

In order to support both binding policies, a rather complex operational semantics, which makes use of two context have to be maintained during a computation: the *global* context and the *partial* context. Accordingly, two context extension operators >> and >>> are provided in Ctx_Prolog. The former, the cactus extension, has a static behavior and extends the partial context while the latter, the linear extension, has a dynamic behavior and extends the global context.

In NemoLOG, we can use dynamic clauses and static clauses to model Ctx_Prolog extended clauses whose body consist of all lazy and eager atoms, respectively.[1] For ex-

[1]Where, however, we use the modal operator [*public*] instead of [*export*].

ample, the Ctx_Prolog clause $\mathtt{p} :- \#\mathtt{q}, \#\mathtt{r}$ corresponds to the dynamic NemoLOG clause $[\mathtt{public}](\mathtt{p} :- \mathtt{q}, \mathtt{r})$, while the clause $\mathtt{p} :- \mathtt{q}, \mathtt{r}$ corresponds to static clause $[public]p :- q, r$. In the case of extension operators, both cactus extension $u >> G$ and linear extension $u >>> G$ are modeled by the NemoLOG goal $[u]G$. The cactus extension can be regarded as the modalized goal $[u]G$ occurring in a static clause, while the linear extension as the modalized goal $[u]G$ occurring in a dynamic clause. Note that in Ctx_Prolog both lazy and eager atoms (linear and cactus extension) are allowed to occur in the same clause body. For instance, $\mathtt{p} :- \#\mathtt{q}, \mathtt{r}$ is a clause. Such a clause cannot be directly represented in NemoLOG because our distinction is made at the level of clauses and not at the level of goals. However, it is sufficient to use two clauses instead of a single one as $[\mathtt{public}](\mathtt{p} :- \mathtt{q}, \mathtt{s})$ and $[\mathtt{public}]\mathtt{s} :- \mathtt{r}$, where $\mathtt{s}$ is a dummy proposition. In this way the subgoal $\mathtt{q}$ is proved dynamically, while $\mathtt{r}$ is proved statically (see also [Giordano and Martelli, 1992]).

Though there is a correspondence between the conservative and evolving policies in Ctx_Prolog and the use of dynamic and static clauses as we have introduced, this correspondence is not perfect, as it is shown by the following example.

Example X.3.5 Let us consider the following Ctx_Prolog and corresponding NemoLOG program:

```
unit(m1) :                  [export][m1]{
a :- d.                         [public]a :- d.
d :- #b.                        [public](d :- b).}

unit(m2) :                  [export][m2]{
b.                              [public]b.}
```

Then, both the goal $\mathtt{m_1} >> \mathtt{m_2} >> \mathtt{d}$ and the goal $\mathtt{m_1} >> \mathtt{m_2} >> \mathtt{a}$ succeed, while it does not in our language, i.e. the goal $[m_1][m_2]d$ succeeds and $[m_1][m_2]a$ fails. In fact, in this case, though the atom $\mathtt{d}$ in the body of the clause $a :- d$ is eager and therefore has to be solved with a clause in $\mathtt{m_1}$, the subgoals generated by it can be solved dynamically. Indeed, the proof of the eager goal $\mathtt{d}$ cam make use of the atom $\mathtt{b}$ defined in the nested module $\mathtt{m_2}$. This behaviour is allowed by means of using two context (the global one and the partial one) instead of a single one as the operational semantics we have defined for NemoLOG (see also [Giordano and Martelli, 1992]).

Chapter XI

Fixed Point Semantics

In this chapter, we present a fixpoint semantics for our language, which is used to prove soundness and completeness of the proof procedure in Section IX.2 with respect to the model theory defined in Section V.2. We also show that there is no loss of generality in restricting first-order Kripke $\mathcal{A}$-interpretations to those in which the domain at each world is the Herbrand universe. It is worth noting that the T_P operator, canonical model construction, and all definitions and proofs are *modular* with respect to the underlying logic of the program specified by means of its set of inclusion axiom clauses.

XI.1 Immediate consequence transformation

We define an *immediate consequence operator* T_P based on a relation of *weak satisfiability* for closed goals in the line of [Miller, 1989a]. This allows to capture the dynamic evolution of the modal context in the operational semantics during a computation.

Completeness with respect to the model theory is proved by a Henkin-style *canonical model* construction, which is similar to the one given in [Bonner *et al.*, 1989].

Interpretations and weak satisfiability

The weak satisfiability is defined on a Kripke-like semantics, where each world represents a modal context and it interprets the program at that modal context. As a result, we define an interpretation for a program P as any function $I : \mathcal{C}^* \to 2^{\mathcal{B}(P)}$; that is a mapping from modal contexts to Herbrand interpretations of the program P. We denote by $\Im$ the set of all interpretations. It is easy to note that $(\Im, \sqsubseteq_\Im)$ is a complete lattice, where $\sqsubseteq_\Im$ is defined as the ordering $I_1 \sqsubseteq_\Im I_2$ if and only if $(\forall\Gamma \in \mathcal{C}^*)$ $I_1(\Gamma) \subseteq I_2(\Gamma)$. The bottom element, denoted by $\perp$, is the interpretation $\perp$ such that $\perp(\Gamma) = \emptyset$, for all context $\Gamma \in \mathcal{C}^*$. Moreover, we define the *join*, denoted by "$\sqcup$", of two interpretations I_1 and I_2 as the interpretation $(I_1 \sqcup I_2)(\Gamma) = I_1(\Gamma) \cup I_2(\Gamma)$, and the *meet*, denoted by "$\sqcap$", of I_1 and I_2 as the interpretation $(I_1 \sqcap I_2)(\Gamma) = I_1(\Gamma) \cap I_2(\Gamma)$.

Definition XI.1.1 (Weak satisfiability) *Let I be an interpretation and let Γ be a modal context, and Ax a set of inclusion axiom clauses then, we say that a closed goal G of* NemoLOG *is* weakly satisfiable *in $I(\Gamma)$, denoted by $I(\Gamma) \mid\models_{Ax} G$, by induction on the structure of G as follows:*

1. *$I(\Gamma) \mid\models_{Ax} T$;*
2. *$I(\Gamma) \mid\models_{Ax} A$ iff $A \in I(\Gamma)$;*
3. *$I(\Gamma) \mid\models_{Ax} G_1 \wedge G_2$ iff $I(\Gamma) \mid\models_{Ax} G_1$ and $I(\Gamma) \mid\models_{Ax} G_2$;*
4. *$I(\Gamma) \mid\models_{Ax} \exists x G'$ iff $I(\Gamma) \mid\models_{Ax} G'[t/x]$, for some $t \in U_P$;*
5. *$I(\Gamma) \mid\models_{Ax} [t]G'$ iff $I(\Gamma') \mid\models_{Ax} G'$, for all $\Gamma' \in \mathcal{C}^*$ such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'$.*

Given an interpretation I and a context Γ, $I(\Gamma) \mid\models_{Ax} G$ means that the goal G is *true* in the interpretation associated with Γ.

Remark XI.1.1 In the rule 5) the derivation relation $\overset{*}{\Rightarrow}_{Ax}$ between modal context depends on the choice of the set of axioms $\mathcal{A}$. A goal $[t]G$ holds in a world Γ if the goal G is true in all worlds reachable from Γ, that is in all world Γ' such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'$. This allows to satisfy the inclusion relation properties of the Kripke $\mathcal{A}$-interpretation which is built by the fixed point semantics (as we will see from the canonical model construction at the page 126).

T_P operator

We are interested in finding an interpretation I such that G is operationally derivable from $\langle Ds, Ax\rangle$ if and only if $I(\varepsilon) \mid\models_{Ax} G$. This particular interpretation is the *least fixed point* of the following immediate consequence transformation T_P defined in the domain of interpretations $(\Im, \sqsubseteq_{\Im})$. We denote with U_P the Herbrand universe of P.

Definition XI.1.2 (Immediate consequence operator) *Let $\langle Ds, Ax\rangle$ be a program of* NemoLOG*, Γ a modal context, and let I be a interpretation, then we define a function T_P from interpretations to interpretations as follows:*

$$\begin{aligned} T_P(I)(\Gamma) \;=\; & \{A \in \mathcal{B}(P) : \Gamma_b(G \supset \Gamma_h A) \in [Ds] \textit{ and } \Gamma_b^*\Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma, \\ & \textit{for some } \Gamma_b^* \textit{ such that } \Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*, \textit{ and } I(\Gamma_b^*) \mid\models_{Ax} G\}. \end{aligned}$$

To prove that T_P is *monotone* and *continuous* we first state two lemmas concerning the weak satisfiability. We present the proof for only the first lemma. The proof of the second is similar.

Lemma XI.1.1 *Given a set Ax of inclusion axiom clauses in* NemoLOG*, if $I_1 \sqsubseteq_{\Im} I_2$ then $I_1(\Gamma) \mid\models_{Ax} G$ implies $I_2(\Gamma) \mid\models_{Ax} G$, for all $\Gamma \in \mathcal{C}^*$.*

Proof. By induction on the structure of G.

$G = T$: Trivial.

$G = A$: If $I_1(\Gamma) \mid\models_{Ax} A$ then, $A \in I_1(\Gamma)$ and, since $I_1 \sqsubseteq I_2$, $A \in I_2(\Gamma)$. Hence $I_2(\Gamma) \mid\models_{Ax} A$.

$G = [t]G'$: If $I_1(\Gamma) \mid\models_{Ax} [t]G'$ then, $I_1(\Gamma') \mid\models_{Ax} G'$ for all modal contexts Γ' such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'$. By inductive hypothesis, $I_2(\Gamma') \mid\models_{Ax} G'$ and, thus, by definition of weak satisfiability, $I_2(\Gamma) \mid\models_{Ax} [t]G$.

$G = G_1 \wedge G_2, G = \exists x G'$: Trivial, from definition of $\mid\models_{Ax}$ applying the inductive hypothesis.

□

Lemma XI.1.2 *Given a set Ax of inclusion axiom clauses in* NemoLOG*, let $I_1 \sqsubseteq_{\Im} I_2 \sqsubseteq_{\Im} I_3 \sqsubseteq_{\Im} \cdots$ be a sequence of interpretations. If G is a goal, $\Gamma \in \mathcal{C}^*$ a modal context and $\bigsqcup_{k\in\omega} I_i(\Gamma) \mid\models_{Ax} G$, then there exist a $k \geq 1$ such that $I_k(\Gamma) \mid\models_{Ax} G$.*

Now we are ready to show that T_P is monotone and continuous.

Theorem XI.1.1 *Given a program $P = \langle Ds, Ax\rangle$ in* NemoLOG*, T_P is monotone, that is, if $I_1 \sqsubseteq I_2$ then $T_P(I_1) \sqsubseteq_{\Im} T_P(I_2)$.*

Proof. Let $I_1 \sqsubseteq_{\Im} I_2$ and assume that $\Gamma \in \mathcal{C}^*$ and $A \in T_P(I_1)(\Gamma)$. Thus, there is a ground clause $\Gamma_b(G \supset \Gamma_h A) \in [Ds]$ such that $\Gamma_b^*\Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, for some Γ_b^* such that $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, and $I(\Gamma_b^*) \mid\models_{Ax} G$. For Lemma XI.1.1 we have that $I_2(\Gamma_b^*) \mid\models_{Ax} G$ and $A \in T_P(I_2)(\Gamma)$. Since Γ and A are arbitrary, we have proved that $T_P(I_1) \sqsubseteq_{\Im} T_P(I_2)$. □

Theorem XI.1.2 *Given a program $P = \langle Ds, Ax\rangle$ in* NemoLOG*, T_P is continuous, that is, if $I_1 \sqsubseteq_{\Im} I_2 \sqsubseteq_{\Im} I_3 \sqsubseteq_{\Im} \ldots$ is a sequence of interpretations, then*

$$\bigsqcup_{k\in\omega} T_P(I_k) = T_P\left(\bigsqcup_{k\in\omega} I_k\right).$$

Proof. We prove the inclusion in two directions.

1. If $I_j \sqsubseteq_{\Im} \sqcup_{k\in\omega} I_k$ for any j, $j \geq 1$, we have, since T_P is monotone, that $T_P(I_j) \sqsubseteq_{\Im} T_P(\sqcup_{k\in\omega}(I_k)$. j is arbitrary, so we can conclude

   $$\sqcup_{k\in\omega} T_P(I_k) \sqsubseteq_{\Im} T_P(\sqcup_{k\in\omega} I_k).$$

2. If $\Gamma \in \mathcal{C}^*$ and $A \in T_P(\sqcup_{k\in\omega} I_k)(\Gamma)$ then there is a ground clause $\Gamma_b(G \supset \Gamma_h A) \in [Ds]$ such that $\Gamma_b^*\Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, for some Γ_b^* such that $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, and $(\sqcup_{k\in\omega} I_k)(\Gamma_b^*) \mid\models_{Ax} G$. By Lemma XI.1.2, there exists a k, $k \geq 1$, such that $T_P(I_k)(\Gamma_b^*) \mid\models_{Ax} G$ and, thus, $A \in T_P(I_k)(\Gamma) \sqsubseteq_{\Im} (\sqcup_{k\in\omega} T_P(I_k)(\Gamma))$. Γ and A are arbitrary, thus

   $$T_P(\sqcup_{k\in\omega}(I_k)) \sqsubseteq_{\Im} \sqcup_{k\in\omega} T_P(I_k).$$

□

The transformation T_P is *monotone* and *continuous* in $(\Im, \sqsubseteq_{\Im})$. Thus, the least fixed point T_P^{ω} of T_P exists by monotonicity and, by continuity, we have $T_P^{\omega}(\bot) = \bigsqcup_{k \in \omega} T_P^k(\emptyset)$, where $T_P^0(\emptyset) = \emptyset$ and, for each $k > 0$, $T_P^k(\emptyset) = T_P(T_P^{k-1}(\emptyset))$.

It is worth noting that for $T_P^{\omega}(\bot)$ the following property holds. It is the fixpoint semantics counterpart of the inclusion property of the Kripke $\mathcal{A}$-interpretations.

Proposition XI.1.1 *Let $P = \langle Ds, Ax \rangle$ be a program, G a closed goal and let Γ be a modal context, then*

$$T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} G \textit{ implies } T_P^{\omega}(\bot)(\Gamma') \mid\models_{Ax} G$$

for all context Γ' such that $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$.

Proof. We prove, by double induction on k and the structure of G, that

$$T_P^k(\bot)(\Gamma) \mid\models_{Ax} G \text{ implies } T_P^k(\bot)(\Gamma') \mid\models_{Ax} G$$

for all $\Gamma' \in \mathcal{C}^*$ such that $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$ and $k \geq 0$.

If $k = 0$ then the theorem holds trivially.

Let assume that the theorem holds for $k-1$ and we prove it for k. We consider the following cases, one for each possible structure of G.

$G = T$: Trivial.

$G = A$: If $T_P^k(\bot)(\Gamma) \mid\models_{Ax} A$ then, $A \in T_P(T_P^{k-1}(\bot))(\Gamma)$. Now, there are two cases:

1. $A \in T_P^{k-1}(\bot)(\Gamma)$. By inductive hypothesis on k, we have that $A \in T_P^{k-1}(\bot)(\Gamma')$, for all $\Gamma' \in \mathcal{C}^*$ such that $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$. Hence, $A \in T_P^k(\bot)(\Gamma')$, for all $\Gamma' \in \mathcal{C}^*$ such that $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$, by Theorem XI.1.1.
2. $A \notin T_P^{k-1}(\bot)(\Gamma)$. There is a clause $\Gamma_b(G' \supset \Gamma_h A) \in [Ds]$ such that $\Gamma_b^* \Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, for some Γ_b^* such that $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, and $T_P^{k-1}(\bot)(\Gamma_b^*) \mid\models_{Ax} G'$. Now, for all Γ', such that $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$, we have that $\Gamma_b^* \Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$. Let us fix a Γ', by inductive hypothesis on k, we have that $T_P^{k-1}(\bot)(\Gamma_b^{*\prime}) \mid\models_{Ax} G'$, for all $\Gamma_b^{*\prime} \in \mathcal{C}^*$ such that $\Gamma_b^* \overset{*}{\Rightarrow}_{Ax} \Gamma_b^{*\prime}$, in particular, since $\Gamma_b^* \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, $T_P^{k-1}(\bot)(\Gamma_b^*) \mid\models_{Ax} G'$. Since $\Gamma_b^* \Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma'$, we have that $A \in T_P^k(\bot)(\Gamma_b^*)$ and, by Definition XI.1.2, $T_P^k(\bot)(\Gamma') \mid\models_{Ax} A$.

$G = [t]G'$: If $T_P^k(\bot)(\Gamma) \mid\models_{Ax} [t]G'$ then, $T_P^k(\bot)(\Gamma^*) \mid\models_{Ax} G'$, for all Γ^* such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma^*$. Now, we have to prove that, $\forall \Gamma'$, $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$, $T_P^k(\bot)(\Gamma') \mid\models_{Ax} [t]G'$, that is, $\forall \Gamma'$, $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$, $\forall \Gamma'^*$, $\Gamma'[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'^*$, $T_P^k(\bot)(\Gamma'^*) \mid\models_{Ax} G'$. Let us fix Γ' and Γ'^*, we have to prove that $T_P^k(\bot)(\Gamma'^*) \mid\models_{Ax} G'$. Since $\Gamma \overset{*}{\Rightarrow}_{Ax} \Gamma'$ and $[t] \overset{*}{\Rightarrow}_{Ax} [t]$, $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'[t]$, therefore $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'^*$ and, by inductive hypothesis, $T_P^k(\bot)(\Gamma'^*) \mid\models_{Ax} G'$.

$G = G_1 \wedge G_2, G = \exists x G'$**:** Trivial, from definition of weak satisfiability applying the inductive hypothesis on the structure.

□

Related work

In [Balbiani *et al.*, 1988] a fixpoint semantics is provided for an instance of MOLOG. In particular, the declarative semantics associated to a program is developed in terms of a *tree*, defined as the fixed point of a certain transformation T_P. Such a tree represents the *minimal Kripke model* of the program. In [Baudinet, 1989] a fixpoint characterization of the declarative semantics of TEMPLOG programs is also given. Both of these languages can be seen to belong to the class of *intensional logic programs* introduced in [Orgun and Wadge, 1992]. There, a *language-independent* theory is developed, which can be applied to a variety of intensional logic programming languages by investigating general properties of *intensional operators* (of which modal operators are a special case). In particular, the authors of [Orgun and Wadge, 1992] use a *neighborhood semantics* of Scott and Montague as an abstract formulation of the denotations of intensional operators and they show that intensional Horn programs (i.e. programs in which atomic formulas can be prefixed by any sequence of intensional operators) have a fixed point characterization of the declarative semantics under some conditions. In particular, intensional operators that appear in clause heads have to be *universal*, *monotonic* and *conjunctive*, and those in clause bodies have to be *monotonic* and *finitary.* Our language *does not belong* to the class of languages that satisfy the above conditions. In fact, the universal modal operators used in clause bodies *are not* finitary. Nevertheless, as we have seen, a fixed point semantics can be given to the language.

XI.2 Soundness and completeness

In this section, we prove the soundness and completeness of fixed point semantics with respect to both operational semantics and possible-worlds semantics. In particular, the correspondence between the fixed point and declarative semantics is proved through a *canonical model construction.*

With respect to operational semantics

The soundness of the fixed point semantics with respect to the operational semantics is given by the following theorem.

Theorem XI.2.1 (Soundness) *Let $P = \langle Ds, Ax\rangle$ be a program of* NemoLOG *and let G be a closed goal, then*

$$T_P^{\omega}(\perp)(\varepsilon) \mid\models_{Ax} G \text{ implies } P, \varepsilon \vdash_o G.$$

Proof. It is proved by showing, with a double induction on k and the structure of G, that $T_P^k(\perp)(\Gamma) \mid\models_{Ax} G$ implies $P, \Gamma \vdash_o G$, for any modal context Γ and $k \geq 0$. In particular, for the empty context ε, we have the main theorem.

If $k = 0$ then the theorem holds trivially.

Let us assume that the theorem holds for $k - 1$ and we prove it for k. We consider the following cases, one for each possible structure of G.

$G = T$: Trivial.

$G = A$: If $T_P^k(\bot)(\Gamma) \mid\models_{Ax} A$ then, $A \in T_P(T_P^{k-1}(\bot))(\Gamma)$. Now, there are two cases:

1. $A \in T_P^{k-1}(\bot)(\Gamma)$. By inductive hypothesis on k, we have that $P, \Gamma \vdash_o A$.
2. $A \notin T_P^{k-1}(\bot)(\Gamma)$. Hence, there is a clause $\Gamma_b(G' \supset \Gamma_h A) \in [Ds]$ such that $\Gamma_b^* \Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, for some Γ_b^* such that $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, and $T_P^{k-1}(\bot)(\Gamma_b^*) \mid\models_{Ax} G'$. By inductive hypothesis on k, we have that $P, \Gamma_b^* \vdash_o G'$ and, by definition of operational derivability, $P, \Gamma \vdash_o A$.

$G = [t]G'$: If $T_P^k(\bot)(\Gamma) \mid\models_{Ax} [t]G'$ then, $T_P^k(\bot)(\Gamma') \mid\models_{Ax} G'$, for all Γ' such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'$. By inductive hypothesis on the structure, $P, \Gamma' \vdash_o G'$, for all Γ' such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'$. In particular, since $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma[t]$, $P, \Gamma[t] \vdash_o G'$, and, by definition of operational derivability, $P, \Gamma \vdash_o [t]G'$.

$G = G_1 \wedge G_2, G = \exists x G'$: Trivial, from definition of operational derivability, applying the inductive hypothesis on the structure.

□

The completeness of fixed point semantics with respect to operational semantics is stated by the following theorem.

Theorem XI.2.2 (Completeness) *Let $P = \langle Ds, Ax \rangle$ be a program of* NemoLOG *and let G be a closed goal, then*

$$P, \varepsilon \vdash_o G \textit{ implies } T_P^{\omega}(\bot)(\varepsilon) \mid\models_{Ax} G.$$

Proof. It is proved by showing, by induction on the length h of the derivation of G, the stronger property that, for any modal context Γ, if $P, \Gamma \vdash_o G$ then $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} G$.

If h is 1 then G must be T then, it is obvious that $P, \Gamma \vdash_o T$ implies $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} T$ by definition of weak satisfiability.

The length of the derivation is $h + 1$. Assume that the theorem holds for derivation with length less or equal than h. We consider the following cases, one for each possible structure of G.

$G = T$: Trivial.

$G = A$: If $P, \Gamma \vdash_o A$ then there exists a clause $\Gamma_b(G' \supset \Gamma_h A) \in [Ds]$ such that $\Gamma_b^* \Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, for some Γ_b^* such that $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, and $P, \Gamma_b^* \vdash_o G'$. By inductive hypothesis, $T_P^{\omega}(\bot)(\Gamma_b^*) \mid\models_{Ax} G'$ and, by Definition XI.1.2, we have that $A \in T_P^{\omega}(\bot)(\Gamma)$. Hence, by definition of weak satisfiability, $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} A$.

$G = [t]G'$: If $P, \Gamma \vdash_o [t]G'$ then $P, \Gamma[t] \vdash_o G'$. Hence, by inductive hypothesis, $T_P^{\omega}(\bot)(\Gamma[t]) \mid\models_{Ax} G'$ and, by Proposition XI.1.1, $T_P^{\omega}(\bot)(\Gamma') \mid\models_{Ax} G'$, for all Γ' such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'$. Thus, by definition of weak satisfiability, we have that $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} [t]G'$.

$G = G_1 \wedge G_2, G = \exists x G'$: Trivial, from the definition of weak satisfiability applying the inductive hypothesis.

□

With respect to possible-worlds semantics

The soundness of fixed point semantics with respect to possible-worlds semantics is stated by the following theorem.

Theorem XI.2.3 (Soundness) *Let $P = \langle Ds, Ax\rangle$ be a program of* NemoLOG*, let G be a closed goal and let Γ be a modal context, then*

$$T_P^{\omega}(\bot)(\varepsilon) \mid\models_{Ax} G \textit{ implies } \models_{\mathcal{A}} Ds \supset G$$

where $\mathcal{A} = \{[t_1]\dots[t_n]\varphi \supset [s_1]\dots[s_m]\varphi \mid [t_1]\dots[t_n] \to [s_1]\dots[s_m] \in Ax\}$.

Proof. We prove, by double induction on k and the structure of G, that $T_P^k(\bot)(\Gamma) \mid\models_{Ax} G$ implies $Ds \models_{\mathcal{A}} \Gamma G$, for any modal context Γ and $k \geq 0$.

If $k = 0$ then the theorem holds trivially.

Let us assume that the theorem holds for $k-1$ and we prove it for k. We consider the following cases, one for each possible structure of G.

$G = T$: Trivial.

$G = A$: If $T_P^k(\bot)(\Gamma) \mid\models_{Ax} A$ then, $A \in T_P(T_P^{k-1}(\bot))(\Gamma)$. Now, there are two cases:

1. $A \in T_P^{k-1}(\bot)(\Gamma)$. By inductive hypothesis on k, we have that $Ds \models_{\mathcal{A}} \Gamma A$.
2. $A \notin T_P^{k-1}(\bot)(\Gamma)$. Hence there is a clause $\Gamma_b(G' \supset \Gamma_h A) \in [Ds]$ such that $\Gamma_b^*\Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, for some Γ_b^* such that $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, and $T_P^{k-1}(\bot)(\Gamma_b^*) \mid\models_{Ax} G'$. Let us assume that $\models_{\mathcal{A}} Ds$, in particular $\models_{\mathcal{A}} \Gamma_b(G' \supset \Gamma_h A)$. Since $\Gamma_b \overset{*}{\Rightarrow}_{Ax} \Gamma_b^*$, $\models_{\mathcal{A}} \Gamma_b^*(G' \supset \Gamma_h A)$, hence $\models \Gamma_b^* G' \supset \Gamma_b^*\Gamma_h A$. Now, by inductive hypothesis on k, $\models_{\mathcal{A}} Ds \supset \Gamma_b^* G'$ and since, by hypothesis, $\models_{\mathcal{A}} Ds$, we have that $\models_{\mathcal{A}} \Gamma_b^*\Gamma_h A$. Finally, since $\Gamma_b^*\Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma$, we have that $\models_{\mathcal{A}} \Gamma A$ holds.

$G = [t]G'$: If $T_P^k(\bot)(\Gamma) \mid\models_{Ax} [t]G'$ then, $T_P^k(\bot)(\Gamma') \mid\models_{Ax} G'$, for all Γ' such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'$. In particular, since $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma[t]$, $T_P^k(\bot)(\Gamma[t]) \mid\models_{Ax} G'$. By inductive hypothesis on the structure, $Ds \models_{\mathcal{A}} \Gamma[t]G'$, that is $Ds \models_{\mathcal{A}} \Gamma G$.

$G = G_1 \wedge G_2, G = \exists x G'$: Trivial, from definition of satisfiability relation and applying the inductive hypothesis on the structure.

□

Let us now consider the completeness of the fixed point semantics with respect to the possible-worlds semantics. The completeness proof is given by constructing a *canonical model* for a given program P, whose domain is constant and is the Herbrand universe U_P of the program.

Definition XI.2.1 (Canonical Model) *The canonical model $\mathcal{M}_c^{Ax}$ for a program $P = \langle Ds, Ax\rangle$ of* NemoLOG *is a tuple $\langle W, \mathcal{R}, D, \mathcal{J}, V\rangle$, where:*

- *$W = \mathcal{C}^*$;*
- *$D = U_P$ (the Herbrand universe of P);*
- *$\mathcal{J}$ is the constant function $\mathcal{J}(w) = U_P$, for all $w \in W$;*
- *V is an assignment function, such that:*
  - *(a) it interprets terms as usual in Herbrand interpretations;*
  - *(b) for each predicate symbol $p \in \mathrm{PRED}^n$ and each world $\Gamma \in \mathcal{C}^*$, $V(p,\Gamma) = \{\langle t_1, \dots, t_n\rangle : T_P^\omega(\bot)(\Gamma) \,||\!\!\models_{Ax} p(t_1, \dots, t_n)$ and $t_1, \dots, t_n \in U_P\}$.*
- *$\mathcal{R}$ is defined as follows: for all $t \in U_P$,*
  $\mathcal{R}_t = \{(\Gamma, \Gamma') \in W \times W : \Gamma, \Gamma' \in \mathcal{C}^*$ and $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'\}$;

The canonical model $\mathcal{M}_c^{Ax}$ for a program P of NemoLOG is a Kripke $\mathcal{A}$-interpretation. In fact, it is easy to see that for each axiom $[t_1]\dots[t_n]\varphi \supset [s_1]\dots[s_m]\varphi$ in $\mathcal{A}$, $\mathcal{R}_{V(t_1)} \circ \dots \circ \mathcal{R}_{V(t_n)} \supseteq \mathcal{R}_{V(s_1)} \circ \dots \circ \mathcal{R}_{V(s_m)}$ holds. It is enough noting that $\mathcal{R}_{V(t_1)} \circ \dots \circ \mathcal{R}_{V(t_n)} = \{(\Gamma, \Gamma') : \Gamma, \Gamma' \in \mathcal{C}^*$ and $\Gamma[t_1]\dots[t_n] \overset{*}{\Rightarrow}_{Ax} \Gamma'\}$, $\mathcal{R}_{V(t_{s_1})} \circ \dots \circ \mathcal{R}_{V(s_m)} = \{(\Gamma, \Gamma') : \Gamma, \Gamma' \in \mathcal{C}^*$ and $\Gamma[s_1]\dots[s_m] \overset{*}{\Rightarrow}_{Ax} \Gamma'\}$ and $[t_1]\dots[t_n] \overset{*}{\Rightarrow}_{Ax} [s_1]\dots[s_m]$.

Completeness proof is based on the following two properties of $\mathcal{M}_c^{Ax}$. They can be proved by induction on the structure of the goals G and the clauses D.

Theorem XI.2.4 *Let $P = \langle Ds, Ax\rangle$ be a program in* NemoLOG*, $\mathcal{M}_c^{Ax}$ its canonical model and let G be a closed goal, then the following properties hold:*

1. *for any $\Gamma \in \mathcal{C}^*$, $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} G$ iff $T_P^\omega(\bot)(\Gamma) \,||\!\!\models_{Ax} G$;*
2. *$\mathcal{M}_c^{Ax}$ satisfies Ds; i.e., for all clauses D in Ds, $\mathcal{M}_c^{Ax}, \varepsilon \models_{\mathcal{A}} D$.*

Where $\mathcal{A} = \{[t_1]\dots[t_n]\varphi \supset [s_1]\dots[s_m]\varphi \mid [t_1]\dots[t_n] \to [s_1]\dots[s_m] \in Ax\}$.

Proof. We prove property 1) by induction on the structure of G.

$G = T$: Trivial.

$G = A$: $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} A$, where $A = p(t_1, \dots, t_n)$, iff $\langle t_1, \dots, t_n\rangle \in V(p, \Gamma)$, that is $T_P^\omega(\bot)(\Gamma) \,||\!\!\models_{Ax} A$.

$G = G_1 \wedge G_2$: $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} G_1 \wedge G_2$ iff $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} G_1$ and $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} G_2$; by inductive hypothesis, $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} G_1$ and $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} G_2$, hence $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} G_1 \wedge G_2$.

$G = \exists x G'$: $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} \exists x G'$ iff $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} G'[t/x]$, for some $t \in U_P$, and, by inductive hypothesis, $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} G'[t/x]$, for some $t \in U_P$, that is $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} \exists x G'$.

$G = [t]G'$: $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} [t]G'$ iff $\mathcal{M}_c^{Ax}, \Gamma' \models_{\mathcal{A}} G'$, for each world Γ' such that $(\Gamma, \Gamma') \in \mathcal{R}_{V(t)}$. By inductive hypothesis and definition of $\mathcal{R}_{V(t)}$, $T_P^{\omega}(\bot)(\Gamma') \mid\models_{Ax} G'$, for each Γ' such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma'$ and hence $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} [t]G'$.

We prove the property 2) if we prove that for all clause D in Ds holds that $\mathcal{M}_c^{Ax}, \varepsilon \models_{\mathcal{A}} D$. First we prove the property that $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} D$, for all clause $\Gamma' D$ in Ds, such that $\Gamma' \overset{*}{\Rightarrow}_{Ax} \Gamma$, with $\Gamma, \Gamma' \in \mathcal{C}^*$. In particular, since ε is a modal context, the main property holds. We prove this property by induction on the structure of D.

$D = G \supset \Gamma_h A$: If $\Gamma'(G \supset \Gamma_h A)$ is in Ds, and $\Gamma' \overset{*}{\Rightarrow}_{Ax} \Gamma$, then $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} G \supset \Gamma_h A$ iff $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} G$ implies $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} \Gamma_h A$. Let us assume $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} G$, by property 1) we have that $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} G$. Hence, by Definition XI.1.2 and since $\Gamma' \overset{*}{\Rightarrow}_{Ax} \Gamma$, $A \in T_P^{\omega}(\bot)(\Gamma^{*\prime\prime})$, that is $T_P^{\omega}(\bot)(\Gamma^{*\prime\prime}) \mid\models_{Ax} A$, for all $\Gamma^{*\prime\prime}$ such that $\Gamma'\Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma^{*\prime\prime}$. Hence, since $\Gamma'\Gamma_h \overset{*}{\Rightarrow}_{Ax} \Gamma'\Gamma_h$ by reflexivity of $\overset{*}{\Rightarrow}_{Ax}$, $T_P^{\omega}(\bot)(\Gamma'\Gamma_h) \mid\models_{Ax} A$. Now, let us assume $\Gamma_h = \Gamma'_h[t]$, if $T_P^{\omega}(\bot)(\Gamma'\Gamma'_h[t]) \mid\models_{Ax} A$, then, by Proposition XI.1.1, $\forall \Gamma^*$, $\Gamma'\Gamma'_h[t] \overset{*}{\Rightarrow}_{Ax} \Gamma^*$, $T_P^{\omega}(\bot)(\Gamma^*) \mid\models_{Ax} A$, hence, by definition of weak satisfiability, $T_P^{\omega}(\bot)(\Gamma'\Gamma'_h) \mid\models_{Ax} [t]A$. We can continue so on until we have that $T_P^{\omega}(\bot)(\Gamma') \mid\models_{Ax} \Gamma_h A$. Again, by Proposition XI.1.1, $\forall \Gamma^*$, $\Gamma' \overset{*}{\Rightarrow}_{Ax} \Gamma^*$, $T_P^{\omega}(\bot)(\Gamma^*) \mid\models_{Ax} \Gamma_h A$, and, in particular, since $\Gamma' \overset{*}{\Rightarrow}_{Ax} \Gamma$, we have that $T_P^{\omega}(\bot)(\Gamma) \mid\models_{Ax} \Gamma_h A$. Finally, by property 1), $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} \Gamma_h A$.

$D = [t]D'$: Let us assume $\Gamma'([t]D')$ is in Ds and $\Gamma' \overset{*}{\Rightarrow}_{Ax} \Gamma$. We have that $\mathcal{M}_c^{Ax}, \Gamma \models_{\mathcal{A}} [t]D'$ if $\mathcal{M}_c^{Ax}, \Gamma^* \models D'$, for all Γ^* such that $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma^*$. By inductive hypothesis, since $(\Gamma'[t])D'$ is in Ds, $\mathcal{M}_c^{Ax}, \Gamma'' \models_{\mathcal{A}} D'$, for any Γ'' such that $\Gamma'[t] \overset{*}{\Rightarrow}_{Ax} \Gamma''$. In particular, since $\Gamma' \overset{*}{\Rightarrow}_{Ax} \Gamma$ and $[t] \overset{*}{\Rightarrow}_{Ax} [t]$, we have that $\Gamma'[t] \overset{*}{\Rightarrow}_{Ax} \Gamma[t]$, $\Gamma[t] \overset{*}{\Rightarrow}_{Ax} \Gamma^*$, that is, by transitivity of the derivation relation, $\Gamma'[t] \overset{*}{\Rightarrow}_{Ax} \Gamma^*$ and, thus, $\mathcal{M}_c^{Ax}, \Gamma^* \models_{\mathcal{A}} D'$.

$D = D_1 \wedge D_2, G = \forall x D'$: Trivial, from the definition of satisfiability relation and applying the inductive hypothesis.

□

By Theorem XI.2.4, the canonical model definition makes it explicit the fact that the fixed point construction builds a Kripke $\mathcal{A}$-interpretation for a program P. We can now prove the following result.

Theorem XI.2.5 (Completeness) *Let $P = \langle Ds, Ax \rangle$ be a program of* NemoLOG*, and let G be a closed goal then,*

$$\models_{\mathcal{A}} Ds \supset G \textit{ implies } T_P^{\omega}(\bot)(\varepsilon) \mid\models_{Ax} G$$

where $\mathcal{A} = \{[t_1] \dots [t_n]\varphi \supset [s_1] \dots [s_m]\varphi \mid [t_1] \dots [t_n] \rightarrow [s_1] \dots [s_m] \in Ax\}$.

Proof. (*If* part) Let us assume that $\models_{\mathcal{A}} Ds \supset G$. Then, for every Kripke $\mathcal{A}$-interpretation $M = \langle W, \mathcal{R}, D, \mathcal{J}, V\rangle$, for every $w \in W$, $M, w \models_{\mathcal{A}} P$ implies $M, w \models_{\mathcal{A}} G$. Hence, in particular, for the canonical model $\mathcal{M}_c^{Ax}$ and the world $\varepsilon \in \mathcal{C}^*$, $\mathcal{M}_c^{Ax}, \varepsilon \models_{\mathcal{A}} P$ implies $\mathcal{M}_c^{Ax}, \varepsilon \models_{\mathcal{A}} G$. By Theorem XI.2.4, property 2), we have that $\mathcal{M}_c^{Ax}, \varepsilon \models_{\mathcal{A}} P$, thus $\mathcal{M}_c^{Ax}, \varepsilon \models_{\mathcal{A}} G$ holds and then, by Theorem XI.2.4, property 1), $T_P^{\omega}(\bot)(\varepsilon) \mid\models_{Ax} G$. $\square$

XI.3 Herbrand domains

In this section we show that for the programs in our language we can, *without loss of generality*, restrict our concern to Kripke interpretations in which the *Herbrand* universe is the constant domain of quantification for each world. A similar property has been proved to hold for other modal and temporal languages and, in particular, for TEMPLOG in [Baudinet, 1989], for an instance of MOLOG in [Balbiani *et al.*, 1988], and for a general class of intensional logic programs in [Orgun and Wadge, 1992]. Moreover, in [Cialdea and Fariñas del Cerro, 1986] a general Herbrand's property has been proved to hold for the modal system Q and, based on it, a first-order extension of propositional modal resolution is defined. In our case this result is a consequence of the completeness and soundness of fixed point semantics with respect to possible-worlds semantics.

In the following we will denote by U_P the Herbrand universe for a program P, i.e., the set of *ground terms* built up from the constants and function symbols that appear in P. Let us start by defining a Kripke interpretation with Herbrand domain.

Definition XI.3.1 *Let P be a program of* NemoLOG. *A Kripke interpretation on the Herbrand universe of P is a Kripke interpretation $M = \langle W, \mathcal{R}, D_H, \mathcal{J}_H, V_H\rangle$ such that:*

- *D_H is the Herbrand Universe of P, U_P;*
- *$\mathcal{J}_H$ is a constant function which maps all worlds in W to the Herbrand universe U_P;*
- *V_H interprets terms as usual in Herbrand interpretations; i.e., $V_H(t) = t$.*

The relation $\models$ between members of W and statements of $\mathcal{I}_{\mathcal{L}}$, the satisfiability, and validity of a closed formula φ of the modal logic is the same of Section V.2. As well as the restriction to the Kripke $\mathcal{A}$-interpretations for characterizing the inclusion modal logics $\mathcal{I}_{\mathcal{L}}^{\mathcal{A}}$.

Theorem XI.3.1 *Let $P = \langle Ax, Ds\rangle$ be a program of* NemoLOG *and G a closed goal, then*

$$\models_{\mathcal{A}} Ds \supset G \text{ if and only if } \models_{\mathcal{A},H} Ds \supset G$$

where $\models_{\mathcal{A},H}$ denotes the satisfiability in Kripke $\mathcal{A}$-interpretations with constant domain U_P and $\mathcal{A} = \{[t_1] \dots [t_n]\varphi \supset [s_1] \dots [s_m]\varphi \mid [t_1] \dots [t_n] \rightarrow [s_1] \dots [s_m] \in Ax\}$.

$$\begin{tikzcd}
 & & Ds \models_{\mathcal{A},H} G \arrow[d, shift right] & & \\
 & & Ds \models_{\mathcal{A}} G \arrow[u, shift right, "\textit{Theorem XI.3.1}"'] \arrow[dl, shift right] \arrow[dr, shift left] & & \\
 & Ds \vdash_{\mathcal{A}} G \arrow[ur, shift right, "\begin{array}{r}\textit{Theorem III.3.1}\\ \textit{Theorem III.3.2}\end{array}"'] \arrow[dl, shift right] & & T^{\omega}_{\langle Ds,Ax\rangle}(\bot)(\varepsilon) \mid\!\models_{Ax} G \arrow[ul, shift left, "\begin{array}{l}\textit{Theorem XI.2.3}\\ \textit{Theorem XI.2.5}\end{array}"'] \arrow[dr, shift left] & \\
Ds \vdash^{u}_{\mathcal{A}} G \arrow[ur, shift right, "\textit{Theorem IX.3.2}"'] \arrow[rrrr, leftrightarrow, "\textit{soundness}", "\textit{completeness}"'] & & & & \langle Ds, Ax\rangle, \varepsilon \vdash_{o} G \arrow[ul, shift left, "\begin{array}{l}\textit{Theorem XI.2.1}\\ \textit{Theorem XI.2.2}\end{array}"']
\end{tikzcd}$$

Figure XI.1: Summary of the results about NemoLOG.

Proof. (*Only if* part) If $\models_{\mathcal{A}} Ds \supset G$ then, $Ds \supset G$ holds in all Kripke $\mathcal{A}$-interpretations. In particular, $Ds \supset G$ is true in all Kripke $\mathcal{A}$-interpretations with constant domain U_P. (*If* part) Let us assume that $\models_{\mathcal{A},H} Ds \supset G$ and $\not\models_{\mathcal{A}} P \supset G$. By Theorem XI.2.3, we have that $T^{\omega}_{P}(\bot)(\varepsilon) \mid\!\not\models_{Ax} G$. Hence, by Theorem XI.2.4, property 1), we have that $\mathcal{M}^{Ax}_{c}, \varepsilon \not\models_{\mathcal{A}} G$. On the other hand, $\mathcal{M}^{Ax}_{c}, \varepsilon \models_{\mathcal{A}} Ds$, by Theorem XI.2.4, property 2), thus, we have that $\mathcal{M}^{Ax}_{c}, \varepsilon \models_{\mathcal{A}} Ds$ and $\mathcal{M}^{Ax}_{c}, \varepsilon \not\models_{\mathcal{A}} G$, i.e. $\mathcal{M}^{Ax}_{c}, \varepsilon \not\models_{\mathcal{A}} Ds \supset G$. Since, by construction, $\mathcal{M}^{Ax}_{c}$ is a Kripke $\mathcal{A}$-interpretation with constant domain U_P, we have that $\mathcal{M}^{Ax}_{c}, \varepsilon \not\models_{\mathcal{A},H} Ds \supset G$, a contradiction. □

Figure XI.1 summarizes the results obtained about NemoLOG.

Chapter XII

Related work

In this part of the thesis, we have developed a modal extension of logic programming that is based on the class of inclusion modal logics. This language (called NemoLOG) is a modal extension of the language of Horn clauses. More precisely, the modal operators may occur in front of clauses, clause heads and in front of goals, and are of the form $[t]$, where t is a term. The properties of the modal operators used in a program, that is the underlying inclusion modal logic that characterizes it, are specified by a set of inclusion clause. Actually, these clauses represent the set $\mathcal{A}$ of inclusion axioms.

Furthermore, we have defined a goal directed proof procedure. Its main advantage is that it is modular with respect to the axiom clauses. This feature is achieved by using a notion of derivation relation between sequences of modalities, which only depends on the properties of modalities themselves. We have also defined a fixpoint semantics by generalizing the standard construction for Horn clauses. It is used to prove soundness and completeness of the operational semantics with respect to model theoretic semantics and it works for the whole class of logics identified by the inclusion axioms. Last but not least, a comparison with the proof theory given in the first part of the thesis is made. In particular, we have shown that in the case of programs and goals of NemoLOG we can restrict to uniform proof as presented in [Miller *et al.*, 1991].

XII.1 Languages based on inclusion modal logics

In [Baldoni *et al.*, 1993] a logic programming language which provides modules as a basic feature is defined. This language is a clausal fragment of an inclusion multimodal logic. In fact, in order to deal with modules, Horn clauses are extended with a collection of modal operators $[m_i]$ of type K. Module composition can be obtained by allowing modules to export clauses or derived facts. To achieve this purpose, a modal operator $\Box$ of type $S4$ is introduced, which makes it possible to distinguish among clauses local to a module, clauses that are fully exported from a module, and those whose consequences are exported. This language allows to model different kinds of modules presented in the literature (so that in each situation the kinds of module that suit better can be adopted). Furthermore, this language provides some well-known features of *object-oriented* programming, like the notion of *self*.

NemoLOG includes the language in [Baldoni *et al.*, 1993] because it is not restricted to a particular inclusion modal logic and because the occurrence of modal operators in front of clauses and clause heads is not restricted to a particular form. Moreover, from the point of view of the programming language it is wider and more flexible. In fact, it allows to define nested and parametric modules and it is possible to represent dependencies among modules in a hierarchy.

In [Baldoni *et al.*, 1997a] an extension of the language in [Baldoni *et al.*, 1993] is presented. In this proposal, both multiple *universal modal operators* and *embedded implications* are allowed. The authors show that this extension is well suited for *structuring knowledge* and, more specifically, for defining *module constructs* within programs for representing *agents beliefs* and, also, for *hypothetical reasoning*. The language is again a clause fragment of the multimodal logic of the proposal in [Baldoni *et al.*, 1993], however, besides the embedded implications, *free* occurrences of modal operators are allowed in front of clauses, clause heads, and goals. In the same way of the language in [Baldoni *et al.*, 1993], a set of modal operators of type K has been used to define modules, by associating a modality with each module for labeling its clauses. In a more general setting, these modalities are used to provide reasoning capabilities in a *multiple agents situation*, by associating a modal operator with each agent to represent its *beliefs*. Moreover, a modal operator $\Box$ of type $S4$ has been used as a weaker version of the common knowledge operator of [Halpern and Moses, 1992].

In [Baldoni *et al.*, 1997a] embedded implications are allowed to occur both in goals and in clause bodies. Languages with *embedded implications* have been extensively studied [Gabbay and Reyle, 1984; Gabbay, 1985; Miller, 1986; McCarty, 1988a; McCarty, 1988b]. These languages allow implications of the form $D \supset G$ which provide a way of introducing *local definitions* of clauses: the clauses in D are intended to be local to the goal G, as they can be used only in a proof of G. The meaning of embedded implications, is that of *hypothetical insertion*: the goal $D \supset G$ is derivable from a program P if G is derivable from the program *updated* with D. When intuitionistic logic is taken as the underlying logic of this language, like in N_Prolog [Gabbay and Reyle, 1984; Gabbay, 1985] and in [Miller *et al.*, 1991], embedded implications allow hypothetical reasoning to be performed, and for this reason they are often called *hypothetical implications*.

In [Giordano *et al.*, 1992; Giordano and Martelli, 1994] the problem of defining structuring facilities in a language with embedded implications is studied. In some way, an embedded implication $D \supset G$ could be compared to what is called a *block* in Algol-like languages, that is, a pair ⟨*definitions, statement*⟩, where D is a set of clause definitions and G plays the role of the statement. In [Giordano *et al.*, 1992] it is shown how different logic languages with embedded implications (or blocks) can be obtained by choosing different *visibility rules* for locally defined clauses. On the other hand, in [Giordano and Martelli, 1994], a modal extension of Horn clause logic (based on the $S4$ logic and, consequently, on an inclusion modal logic) is defined to provide a unifying framework in which these different kinds of local definitions of clauses can be integrated. These extensions with embedded implications provide different notions of a block, from which various kinds of modules can also be defined, by introducing some syntactic sugar.

In [Baldoni *et al.*, 1997a] embedded implications have been used to introduce *local* definitions of clauses like *blocks* in imperative programming and for performing some form of *hypothetical reasoning.* In particular, since $\Box$ is an $S4$ modality, the language subsumes N_Prolog: by adopting the well-known translation of intuitionistic logic to modal logic $S4$, N_Prolog clauses can be translated in this language.

The logic programming language presented in [Baldoni *et al.*, 1997a] is modal logic refinement of *hereditary Harrop formulae* [Miller *et al.*, 1991], and it lies on the same line as other logic programming languages which are not based on classical first-order logic, like, for instance, those based on intuitionistic logic [Gabbay and Reyle, 1984; Gabbay, 1985; McCarty, 1988a; McCarty, 1988b; Miller, 1986; Miller, 1989a; Miller, 1989b], higher-order logic [Miller *et al.*, 1991], and linear logic [Hodas and Miller, 1991].

In [Baldoni *et al.*, 1996b] a translation to standard Horn clauses for the language in [Baldoni *et al.*, 1997a] is presented, this translation method consisting of two steps. In the first step all embedded implications are eliminated so to obtain a program consisting only of modal Horn clauses. This step requires the introduction of a new modal operator for each embedded implication, so that the extracted clauses can be used only in the right environment. The second step is based on an approach similar to the functional translation: modalities are eliminated by adding to each predicate an argument which represents the modal context.

Despite the fact that NemoLOG does not allow embedded implications to occur in goals and in the body of clauses, it can deal with modal operators which are characterized by more complex properties than the ones in [Baldoni *et al.*, 1997a]. Due to the fact that it is possible to define "ad hoc" interaction axioms by means of the inclusion axiom clauses, NemoLOG can express more sophisticated knowledge information and, then, it is better suited to perform epistemic reasoning. Furthermore, as we have seen in Chapter X, inclusion axiom clauses allow to define hierarchical dependencies among modules in a simple way.

In [Baldoni *et al.*, 1993; Baldoni *et al.*, 1997a], the logic is described by defining a sequent calculus for it. The sequent calculus is used to prove soundness and completeness of the proof procedures with respect to the model theoretic semantics showing that the proof procedure looking for derivations which correspond to sequent proofs of a certain form. The approach could be regarded as being complementary to the one in [Giordano and Martelli, 1994]. In fact, in [Giordano and Martelli, 1994], the soundness and completeness of the proof procedure with respect to the Kripke semantics has been proved by making use of a Henkin-style canonical model construction. In [Baldoni *et al.*, 1993; Baldoni *et al.*, 1997a], instead, the goal directed proof procedures is proved sound and complete by means of a sequent calculus.

If we compare the kind of sequent proofs in [Baldoni *et al.*, 1993; Baldoni *et al.*, 1997a] with *uniform proofs* as presented in [Miller *et al.*, 1991], we can observe that the former are not uniform. As already remarked in Chapter IX, this happens because a *prefixed* sequent calculus is not used. In fact, since NemoLOG subsumes the language in [Baldoni *et al.*, 1993] and that we have proved that for programs and goals in NemoLOG there exists a notion of uniform proof, we have that a notion of uniform proof exists also for the language in [Baldoni *et al.*, 1993]. Furthermore, we believe that a similar proof could be given also for the language in [Baldoni *et al.*, 1997a] provided that the prefixed sequent calculus like the one here adopted is used.

In [Baldoni *et al.*, 1996a], it is presented a *framework* for developing modal extensions of logic programming, which is parametric with respect to the properties chosen for the modal operators and which allow sequences of universal modalities to occur in front of clauses, goals, and clause heads. This work is at the basis of our logic programming language NemoLOG.

Finally, we would like to mention the modal programming language $\mathcal{L}^{\mathcal{A}}$ for reasoning about actions presented in [Baldoni *et al.*, 1997b]. $\mathcal{L}^{\mathcal{A}}$ makes use of abductive assumptions to deal with *persistence* and provides a solution to the *ramification* problem by allowing one-way "causal rules" to be defined among fluents. Both the semantics and the goal directed abductive proof procedure are defined within the argumentation framework [Bondarenko *et al.*, 1993; Dung, 1993b] developing a three-valued semantics which can be regarded as a generalization of Dung's admissibility semantics [Dung, 1993a] to modal settings.

The language $\mathcal{L}^{\mathcal{A}}$ can be regarded as an extension of Gelfond and Lifschitz' language $\mathcal{A}$ [Gelfond and Lifschiftz, 1993]. However, rather than following the way of defining a language with an "ad hoc" (and high-level) semantics and, then, translating it into a logic programming language with negation as failure, in [Baldoni *et al.*, 1997b] actions are represented by modal operators and the semantics is a standard Kripke semantics. The reason is that modal logics allow to interpret actions as state transitions through the accessibility relations in a natural way.

XII.2 Other languages

NemoLOG bears strong similarities with MOLOG [Fariñas del Cerro, 1986] (later evolved in TIM [Balbiani *et al.*, 1991]), a framework for modal logic programming in which the user can fix the underlying modal logic. In MOLOG both *existential* and *universal* modalities can occur in front of clauses, in front of clause heads, and in front of goals. A resolution procedure (close to Prolog resolution) is defined for modal Horn clauses in the logic $S5$ which contains only universal modal operators of the form $knows(t)$, where t is an arbitrary term. TIM is a *meta-level* inference system which can support some well-known modal systems and epistemic logics such as Q, T, $S4$, and $S5$ and it provides a general methodology to implement non-classical logics. Though the language in similar to ours, the properties of $S5$ modalities are different from the ones we have considered, in the sense that we did not take into account $S5$. In [Balbiani *et al.*, 1988], instead, a modal SLD-resolution method is presented for a fragment of MOLOG in which $\Box$ cannot be used in the bodies of modal clauses (while $\Diamond$ can). Some different modal systems (Q, T and $K4$) are considered. A fixpoint semantics is also provided.

Modal logic programming languages based on $S5$ have been also proposed in [Akama, 1986]. There, a program is defined as a set of modal definite clauses whose literals are prefixed by any sequence of universal and existential modalities. An SLD-resolution procedure is defined for these languages.

In NemoLOG universal modalities are allowed to freely occur in front of clauses, clause heads and clause bodies (or goals), while existential modal operators are not allowed. In particular, differently than other languages proposed in the literature, like TEMPLOG [Abadi and Manna, 1989], Temporal Prolog [Gabbay, 1987], the fragment of MOLOG in [Balbiani *et al.*, 1988], and the language in [Akama, 1986], existential modalities are not allowed to occur in front of goals. In spite of this limitation, the features of parametric modalities and the possibility of introducing inclusion axioms, make NemoLOG well suited for performing some epistemic reasoning, for defining parametric and nested modules, for representing inheritance in a hierarchy of classes and for reasoning about actions.

Actually, NemoLOG could be extended to allow existential modalities in front of goals. Indeed, due to the analogy between universal (existential) quantifiers and universal (existential) modalities, and from the fact that, in standard logic programs, universal quantifiers occur in front of clauses, while existential quantifiers occur in front goals, the use of existential modalities should be possible. Of course, to deal with existential modalities $\langle t\rangle$ in front of goals, the proof procedure presented in Chapter IX should be modified substantially. The main difference would be that, since existential modalities $\langle t\rangle$ do not distribute on conjunctions, a goal $\langle t\rangle(G_1 \wedge G_2)$ cannot be proved by proving the two subgoals $\langle t\rangle G_1$ and $\langle t\rangle G_2$. For this reason, the policy of recording the sequence of modalities that are found in front of a goal in a context Γ does not work in that case in a straightforward way.

TEMPLOG is a temporal logic programming language and it allows temporal operators like $\bigcirc$ (*next moment in time*), $\Box$ (*from now on*), and $\Diamond$ (*sometime in the future*) to occur in Horn clauses. $\Diamond$ is allowed in front of goals while $\Box$ is not. In our language, while existential modalities are not admitted, universal modalities can occur in goals and clause bodies. Despite these differences, there are some similarities with TEMPLOG. In particular, in TEMPLOG a distinction is made between *initial* clauses ($G \supset A$ and $G \supset \Box A$), and *permanent* clauses ($\Box(G \supset A)$). This distinction is quite similar to ours between *local*, *static* and *dynamic* clauses (see Chapter X).

Temporal Prolog [Gabbay, 1987] allows occurrences of temporal operators like F (*sometime in the future*), P (*sometime in the past*), $\Box$ (*always*). This language is rather different from ours and in particular, it admits embedded implications in clause heads.

We have already mentioned to the *translation* approach to modal logics in Chapter VII. In the case of modal logic programming, this approach has been used in [Debart *et al.*, 1992; Nonnengart, 1994] to obtain a standard Prolog program starting form Horn clauses extended with modal operators. In [Debart *et al.*, 1992] the functional translation method is extended to multimodal logic and it is applied to modal logic programming. The modalities considered are both universal and existential, and are of any type among KD, KT, $KD4$, $KT4$, KF. Interaction axioms of the form $I(a_i, a_j) : [a_i]\varphi \supset [a_j]\varphi$ are allowed but, in [Debart *et al.*, 1992], general inclusion axioms as the ones in NemoLOG are not considered. Nonnengart has proposed a mixed approach based on a relational and functional translation [Nonnengart, 1993]. One of his aims is to avoid theory unification. As a particular case, following this approach, modal Horn clauses can be directly translated to Prolog clauses [Nonnengart, 1994]. This method requires that accessibility relation properties are first-order predicate logic definable. Moreover, if Prolog is to be used as a first order inference machine, accessibility relation properties must be defined through Horn clauses. In particular, he can provide Prolog translation for modalities with the properties of KD, KT, $KD4$, $S4$ and he can also deal with axioms like $(B) : \varphi \supset \Box\Diamond\varphi$, and, hence, with logics like KDB, $KD45$ and $S5$.

An optimization of the functional translation method for the class of inclusion logics has been proposed in [Gasquet, 1993], where, however, seriality is assumed for each modal operator. Then, since we deal with modal Horn clauses containing only universal modalities, the case we consider can be regarded as a special instance of the one in [Gasquet, 1993]. In particular, in the case when only ground terms can occur within modalities in the program, in the goal and in the axioms (which is the one he considers), the generality of equational unification may be replaced with a notion of matching (or a notion of string rewriting). Differently than [Gasquet, 1993], we deal with parametric modalities. However, in the general case when variables occur within modalities we also need some form of equational unification.

Conclusions

In this thesis we have studied the class of normal multimodal logics determined by axiom schemas of the form

$$[t_1][t_2]\dots[t_n]\varphi \supset [s_1][s_2]\dots[s_m]\varphi \quad (n > 0, m \geq 0)$$

This class is called *inclusion modal logics* because it is characterized by particular inclusion properties between accessibility relations. For this class of logics we have defined a *prefixed analytic tableau calculus* and given some *undecidability* and *decidability* results. First-order is also considered, though only in the case of increasing domains.

Afterwards, we have extended the class of the considered multimodal logics and, in particular, we have focused on the ones that are characterized by axiom schemas of the form

$$G^{a,b,c,d} : \langle a\rangle[b]\varphi \supset [c]\langle d\rangle\varphi$$

where the labels of the modal operators are arbitrarily complex parameters, built from the atomic ones, by using an operator of composition and an operator of union. The incestual axiom copes with most of the well-known axioms, such as T, D, B, 4, 5, and their multimodal versions. For this class of logics we have introduced a tableau calculus that is a generalization of the one presented for the inclusion modal logics.

In the course of this work, we have also defined a logic programming language based on the above class of inclusion modal logics. This language, called NemoLOG, extends the Horn clause language by allowing *free* occurrences of universal modal operators in front of clauses, in front of clause heads, and in front of goals. NemoLOG is *parametric* with respect to the class of inclusion modal logics and this feature is achieved by adding to a program a collection of *inclusion axiom clauses* of the form $[t_1]\dots[t_n] \rightarrow [s_1]\dots[s_m]$, one for each inclusion axiom schema of the considered logic.

NemoLOG is particularly suitable to represent *knowledge* and *beliefs* of agents. Moreover, due to the fact that we can characterize our modal operators by means of arbitrary inclusion axioms, our language is particularly well-suited to performing epistemic reasoning in a *multiagent situation* with *interactions* between agents. Moreover, in a *software engineering* settings, we have shown how to use NemoLOG to *modularize* logic programs in order to enhance their readability and reusability; *parametric* and *nested* modules are considered. Furthermore, NemoLOG allows to define *hierarchies* among modules and *inheritance* mechanisms similar to the ones of object-oriented languages.

NemoLOG has a *goal directed proof procedure* which is *modular* with respect to the properties of modalities: it uses a notion of derivation relation between sequences of modal operators, which only depends on the properties of modalities themselves. As it is usual in logic programming setting, a fixed point semantics is given and it is used to prove the soundness and the completeness of the proof procedure with respect to model theoretic semantics.

Indeed, despite the fact that NemoLOG shows quite a simple operational semantics, where the properties of the modal operators are factored out by means of a derivation relation, we think that it is better to consider the NemoLOG language as a *framework* for developing modal extension of logic programming aimed at solving particular problems. The examples shown in Chapter X can actually be reconsidered in this perspective: on one hand, restricting to specific cases it is possible to improve the language itself for the case at issue, optimizing at the same time the computational aspects, while on the other hand the general framework supplies theoretical results that can be inherited by the specific case studies. For instance, in the case of the problem of dealing with inheritance in hierarchies of modules and, in a more general setting, with the introduction of object-oriented features in a logic programming language, tackled in Chapter X, only a few axiom kinds were used. It would be interesting to deepen this investigation by making the proof procedure effective, i.e. to see if the general procedure, which cannot be implemented in an easy way, can be operationalized for the case of interest. Another example is the restriction of the language to dealing with actions and change. We are currently working at the development of a specialization of the framework for reasoning about dynamic domains in a logic programming setting [Baldoni *et al.*, 1997b; Baldoni *et al.*, 1998b]. To summarize, we think that it is important to consider Logic Programming as a general framework that supplies proof theories and other theoretical results that, specialized or extended ad hoc for the particular application, can be exploited for building languages and systems that solve a broad variety of problems.

Nevertheless, the work presented in this thesis is in progress and lots of problems are still open.

We have shown that the class of right-regular modal logics is decidable, however, we say nothing about the decidability of the inclusion modal logics based on *left* type-0 grammars, i.e. grammars whose production rules are of the form $A \to A'\sigma$ or $A \to \sigma$, where A, A' are variables and σ is a string of terminals. We believe that also this class is decidable but the technique used to prove the decidability for the right-regular modal logics does not work for it.

Another open problem regards a decision procedure. Apart from the naive algorithm given by generating all finite Kripke interpretations for checking whether or not a formula is a theorem, we have seen that our tableau calculus is not a decision procedure even if it deals with decidable modal logics. It would be interesting to transform it in a decision procedure, in the line of the works in [Fitting, 1983; Massacci, 1994].

In Chapter VI, we have introduced complex parameters as labels for the modal operators. In this case, we have used an operator of composition and an operator of union, however, further extensions could also be incorporated. For example, we could add the *iteration* operator "$*$" and the *test* operator "?" of dynamic logic to the language. In this case two questions would arise: how to extend the tableau calculus in order to deal with these new operators? And, moreover, what is the relationship among multimodal logics and dynamic logics? A recent work [De Giacomo and Massacci, 1996] shows a tableau calculus for dynamic logic that could be at the basis for such a kind of study.

As remarked in [Fitting, 1996], although *resolution* is the most used approach to automated deduction, tableaux will continue to have a great importance because they are relatively easy to develop due to the strong relationship with the semantics issue, i.e. they rely on the *explicit* construction of models. We think it would also be a goal of this research to develop a theorem prover for better studying the expressive features of the logics considered; this would also help to implement a NemoLOG inference machine.

We have seen, in Chapter IV, that the *validity problem* for the whole class of inclusion modal logic is undecidable and it still remains undecidable even though we restrict our attention to some subclasses. However, we have not studied what happens restricting to modal Horn clauses of NemoLOG. For example, the class of inclusion modal logics based on context-free grammars is undecidable but the method used to prove this in Theorem IV.2.2 does not work in the case of formulae of NemoLOG.

Furthermore, another important problem related to our logic programming language that we have not studied is the *computational complexity* of the satisfiability. On the other hand, when we do not consider the multimodal case, our definition of modal Horn clauses falls into the one given in [Fariñas del Cerro and Penttonen, 1987; Chen and Lin, 1994] where the problem of the complexity of the satisfiability of modal Horn clauses is studied for different modal logics. In particular, in [Chen and Lin, 1994], it is shown that the satisfiability problem of modal Horn clauses for each of K, T, and $S4$ is PSPACE-complete.

Finally, it is worth noting that in Chapter IX we have proved that for programs and goals of NemoLOG there exists a uniform proof in the prefixed sequent calculus that we have defined. However, we do not know if such a kind of proof exists also for fragments of inclusion modal logics that are wider than the clausal fragment we have given. In particular, we refer to the possibility of extending our language allowing free occurrences of *embedded implications* in goals and clause bodies in the line of [Giordano and Martelli, 1994; Baldoni *et al.*, 1997a]. The existence of a uniform proof would be a powerful tool to study a *goal directed* proof procedure for the extended language.

Appendix A

Some examples of translated NemoLOG programs

In this appendix we present some example of NemoLOG programs translated into standard Horn clause logic obtained applying the translation method defined in Section IX.4.

Before presenting the programs, it is necessary to give some more information. In particular, we have represented a sequence of modal operators by means of the list of labels of the modalities themselves. For example, the sequence $[animal][bird][tweety]$ is represented by [`animal`, `bird`, `tweety`]. Consequently, the operator "•" defined at page 98 is simply implemented by the predicate `append/3`, the concatenation relation for lists:

```
% The concatenation relation for lists.

append([], L2, L2).
append([X | L1], L2, [X | L]):-
        append(L1, L2, L).
```

In this way a translated clause would be of the form:

$$\begin{array}{rcl} A_0(X) & :- & derive(\Gamma_b, \Gamma_h, X, Y), \\ & & append(Y, \Gamma_{g_1}, Y_{g_1}), A_1(Y_{g_1}), \\ & & \ldots, \\ & & append(Y, \Gamma_{g_m}, Y_{g_m}), A_m(Y_{g_m}) \end{array}$$

However, in order to simplify the form of a translated clause, we have chosen to move the `append` at the top of a predicate definition, before the call of the predicate *derive*/4. To do so we need to add another argument to all predicate definition:[1]

$$A_0(X_1, X_2) :- append(X_1, X_2, X), derive(\Gamma_b, \Gamma_h, X, Y), A_1(Y_{g_1}, \Gamma_{g_1}), \ldots, A_m(Y_{g_m}, \Gamma_{g_m})$$

The following predicate definitions are common to all programs.

[1]Note that, in the programs presented in the following we use the predicate `derive1/5` which performs the concatenation and derivation operations together.

```
% The membership relation.

member(X, [X | _]):-!.
member(X, [_ | List]):-
        member(X, List).

% The prefix relation. The element that forms the prefix
% must be among the ones which belong to a given set.

prefix_in([], _, _).
prefix_in([X | Prefix], [X | List], Set):-
        member(X, Set),
        prefix_in(Prefix, List, Set).

% The predicate derive/5 performs the concatenation of the
% lists X1 and X2 and the derivation operation returning
% the new context in Y.

derive1(Gamma_b, Gamma_h, X1, X2, Y):-
        append(X1, X2, Gamma),
        derive(Gamma_b, Gamma_h, Gamma, Y).
```

For each example in the following we show the “ad hoc” `derive/4` predicate and the translated program.

Program A.1 : Fibonacci numbers.

```
derive([], [], [], []).
derive([], [always | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [always, next]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [next | Gamma_h], [next | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([always | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [always, next]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([next | Gamma_b], Gamma_h, [next | Gamma], [next | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).


fib(X1, X2, 0):-
        derive1([], [], X1, X2, _).
fib(X1, X2, 1):-
        derive1([], [next], X1, X2, _).
```

```
fib(X1, X2, A):-
        derive1([always], [next, next], X1, X2, Y),
        fib(Y, [], B),
        fib(Y, [next], C),
        A is B + C.
```

Program A.2 : Friends puzzle I and II.

```
derive([], [], [], []).
derive([peter], [john | Gamma_h], Gamma, [peter]):-
        prefix_in(Prefix, Gamma, [peter, john]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([wife(peter)], [john | Gamma_h], Gamma, [peter]):-
        prefix_in(Prefix, Gamma, [wife(peter), peter, john]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [peter, john | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [peter, john]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [fool | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [fool, wife(peter), peter, john]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [wife(peter), john | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [wife(peter), peter, john]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [wife(peter) | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [wife(peter), peter]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [peter | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [peter]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [john | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [john]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([peter, john | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [peter, john]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([wife(peter), john | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [wife(peter), peter, john]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([fool | Gamma_b], Gamma_h, Gamma, NResult):-
```

```
        prefix_in(Prefix, Gamma, [fool, wife(peter), peter, john]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([wife(peter) | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [wife(peter), peter]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([peter | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [peter]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([john | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [john]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).


time(X1, X2):-
        derive1([], [peter], X1, X2, _).
time(X1, X2):-
        derive1([wife(peter)], [john], X1, X2, Y),
        time(Y, [peter]).
place(X1, X2):-
        derive1([], [peter, john], X1, X2, _).
appointment(X1, X2):-
       derive1([peter, john], [], X1, X2, Y),
       place(Y, []),
       time(Y, []).

% In the case of Friends puzzle II the relation appointment
% is defined by the following one:

appointment(X1, X2):-
        derive1([fool], [], X1, X2, Y),
        place(Y, []),
        time(Y, []).
```

Program A.3 : Bubblesort I.

```
derive([], [], [], []).
derive([], [export | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [export, list, sort]),
        append(Prefix, Suffix, Gamma),
derive([], [list | Gamma_h], [list | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [sort | Gamma_h], [sort | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([export | Gamma_b], Gamma_h, Gamma, NResult):-
```

```
        prefix_in(Prefix, Gamma, [export, list, sort]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([list | Gamma_b], Gamma_h, [list | Gamma], [list | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([sort | Gamma_b], Gamma_h, [sort | Gamma], [sort | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).


% module list.

new_append(X1, X2, [], X, X):-
        derive1([export, list], [], X1, X2, _).
new_append(X1, X2, [A | B], C, [A | B1]):-
        derive1([export, list], [], X1, X2, Y),
        new_append(Y, [], B, C, B1).

% module sort.

busort(X1, X2, L, S):-
        derive1([export, sort], [], X1, X2, Y),
        new_append(Y, [list], C, [A, B | D], L),
        B < A, !,
        new_append(Y, [list], C, [B, A | D], M),
        busort(Y, [], M, S).
busort(X1, X2, S, S):-
        derive1([export, sort], [], X1, X2, _).
```

Program A.4 : Bubblesort II.

```
derive([], [], [], []).
derive([], [export | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [export, list, sort]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [list | Gamma_h], [list | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [sort | Gamma_h], [sort | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([export | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [export, list, sort]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([list | Gamma_b], Gamma_h, [list | Gamma], [list | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([sort | Gamma_b], Gamma_h, [sort | Gamma], [sort | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).


% module list.
```

```
new_append(X1, X2, [], X, X):-
        derive1([export, list], [export], X1, X2, _).
new_append(X1, X2, [A | B], C, [A | B1]):-
        derive1([export, list], [export], X1, X2, Y),
        new_append(Y, [], B, C, B1).

% module sort.

busort(X1, X2, L, S):-
        derive1([export, sort], [export], X1, X2, Y),
        new_append(Y, [list], C, [A, B | D], L),
        B <  A, !,
        new_append(Y, [list] ,C, [B, A  |D], M),
        busort(Y, [], M, S).
busort(X1, X2, S, S):-
        derive1([export, sort], [export], X1, X2, _).
```

Program A.5 : Bubblesort III and IV.

```
derive([], [], [], []).
derive([], [export | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [export, list, sort(_),
                                  ascending, descending, cartesian(_, _)]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [list | Gamma_h], [list | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [ascending | Gamma_h], [ascending | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [descending | Gamma_h], [descending | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [sort(X) | Gamma_h], [sort(X) | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [cartesian(X, Y) | Gamma_h], [cartesian(X, Y) | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([export | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [export, list, sort(_),
                                  ascending, descending, cartesian(_, _)]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([list | Gamma_b], Gamma_h, [list | Gamma], [list | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([ascending | Gamma_b], Gamma_h, [ascending | Gamma], [ascending | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([descending | Gamma_b], Gamma_h, [descending | Gamma], [descending | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([sort(X) | Gamma_b], Gamma_h, [sort(X) | Gamma], [sort(X) | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([cartesian(X, Y) | Gamma_b], Gamma_h, [sort(X) | Gamma], [cartesian(X, Y) | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
```

```
% module list.

new_append(X1, X2, [], X, X):-
        derive1([export, list], [], X1, X2, _).
new_append(X1, X2, [A | B], C, [A | B1]):-
        derive1([export, list], [], X1, X2, Y),
        new_append(Y, [], B, C, B1).

% module ascending.

ordered(X1, X2, A, B):-
        derive1([export, ascending], [], X1, X2, _),
        A < B.

% module descending.

ordered(X1, X2, A, B):-
        derive1([export, descending], [], X1, X2, _),
        A > B.

% module cartesian(Ord1, Ord2).

ordered(X1, X2, [A, B], [U, V]):-
        derive1([export, cartesian(Ord1, Ord2)], [], X1, X2, Y),
        ordered(Y, [Ord1], A, U).
ordered(X1, X2, [A, B], [A, V]):-
        derive1([export, cartesian(Ord1, Ord2)], [], X1, X2, Y),
        ordered(Y, [Ord2], B, V).

% module sort(Order).

busort(X1, X2, L, S):-
        derive1([export, sort(Order)], [], X1, X2, Y),
        new_append(Y, [list], C, [A, B | D], L),
        ordered([export, sort(Order)], [Order], B, A),
        new_append(Y, [list], C, [B, A | D], M),
        busort(Y, [], M, S).
busort(X1, X2, S, S):-
        derive1([export, sort(Order)], [], X1, X2, _).
```

Program A.6 : Animal taxonomy I.

```
derive([], [], [], []).
derive([],[export | Gamma_h], Gamma, []):-
        prefix_in(Prefix, Gamma, [export, animal, horse, bird, tweety]),
        append(Prefix, Suffix, Gamma),
        derive([], Gamma_h, Suffix, _).
derive([], [animal | Gamma_h], [animal | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [horse | Gamma_h], [horse | Gamma], []):-
```

```
        derive([], Gamma_h, Gamma, _).
derive([], [bird | Gamma_h], [bird | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [tweety | Gamma_h], [tweety | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([export | Gamma_b], Gamma_h, Gamma, NResult):-
        prefix_in(Prefix, Gamma, [export, animal, horse, bird, tweety]),
        append(Prefix, Suffix, Gamma),
        derive(Gamma_b, Gamma_h, Suffix, Result),
        append(Prefix, Result, NResult).
derive([animal | Gamma_b], Gamma_h, [animal | Gamma], [animal | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([horse | Gamma_b], Gamma_h, [horse | Gamma], [horse | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([bird | Gamma_b], Gamma_h, [bird | Gamma], [bird | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).
derive([tweety | Gamma_b], Gamma_h, [tweety | Gamma], [tweety | Result]):-
        derive(Gamma_b, Gamma_h, Gamma, Result).


% class animal.

mode(X1, X2, walk):-
        derive1([animal, export], [], X1, X2, _).
mode(X1, X2, run):-
        derive1([animal, export], [], X1, X2, Y),
        no_of_legs(Y, [], X),
        X >= 2.
mode(X1, X2, gallop):-
        derive1([animal, export], [], X1, X2, Y),
        no_of_legs(Y, [], X),
        X >= 4.

% class horse.

no_of_legs(X1, X2, 4):-
        derive1([animal, horse, export], [], X1, X2, _).
covering(X1, X2, hair):-
        derive1([animal, horse, export], [], X1, X2, _).

% class bird.

no_of_legs(X1, X2, 2):-
        derive1([animal, bird, export], [], X1, X2, _).
covering(X1, X2, feather):-
        derive1([animal, bird, export], [], X1, X2, _).
mode(X1, X2, fly):-
        derive1([animal, bird, export], [], X1, X2, _).

% class tweety.
```

```
owner(X1, X2, fred):-
        derive1([animal, bird, tweety, export], [], X1, X2, _).
```

Program A.7 : Animal taxonomy II and Humans.

```
derive([], [], [], []).
derive([], [animal | Gamma_h], [X | Gamma], []):-
        member(X, [animal, horse, bird, tweety, human(_, _), peter, jane, john]),
        derive([], Gamma_h, Gamma, _).
derive([], [bird | Gamma_h], [X | Gamma], []):-
        member(X, [bird, tweety]),
        derive([], Gamma_h, Gamma, _).
derive([], [horse | Gamma_h], [horse | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [tweety | Gamma_h], [tweety | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [human(S, A) | Gamma_h], [human(S, A) | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [human(male, 30) | Gamma_h], [X | Gamma], []):-
        member(X, [human(male, 30), peter]),
        derive([], Gamma_h, Gamma, _).
derive([], [human(female, 42) | Gamma_h], [X | Gamma], []):-
        member(X, [human(female, 42), jane]),
        derive([], Gamma_h, Gamma, _).
derive([], [human(male, 45) | Gamma_h], [X | Gamma], []):-
        member(X, [human(male, 45), john]),
        derive([], Gamma_h, Gamma, _).
derive([], [mathematician | Gamma_h], [X | Gamma], []):-
        member(X, [mathematician, john]),
        derive([], Gamma_h, Gamma, _).
derive([], [peter | Gamma_h], [peter | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [jane | Gamma_h], [jane | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([], [john | Gamma_h], [john | Gamma], []):-
        derive([], Gamma_h, Gamma, _).
derive([animal | Gamma_b], Gamma_h, [X | Gamma], [X | Result]):-
        member(X, [animal, horse, bird, tweety, human(_, _), peter, jane, john]),
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([bird | Gamma_b], Gamma_h, [X | Gamma], [X | Result]):-
        member(X, [bird, tweety]),
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([horse | Gamma_b], Gamma_h, [horse | Gamma], [horse |  Result]):-
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([tweety | Gamma_b], Gamma_h, [tweety | Gamma], [tweety | Result]):-
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([human(S, A) | Gamma_b], Gamma_h, [human(S, A) | Gamma], [human(S, A) | Result]):-
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([human(male, 30) | Gamma_b], Gamma_h, [X | Gamma], [X | Result]):-
        member(X, [human(male, 30), peter]),
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([human(female, 42) | Gamma_b], Gamma_h, [X | Gamma], [X | Result]):-
```

```
        member(X, [human(female, 42), jane]),
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([human(male, 45) | Gamma_b], Gamma_h, [X | Gamma], [X | Result]):-
        member(X, [human(male, 45), john]),
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([mathematician | Gamma_b], Gamma_h, [X | Gamma], [X | Result]):-
        member(X, [mathematician, john]),
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([peter | Gamma_b], Gamma_h, [peter | Gamma], [peter | Result]):-
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([jane | Gamma_b], Gamma_h, [jane | Gamma], [jane | Result]):-
        derive(Gamma_b, Gamma_h, Gamma,  Result).
derive([john | Gamma_b], Gamma_h, [john | Gamma], [john | Result]):-
        derive(Gamma_b, Gamma_h, Gamma,  Result).


% class animal.

mode(X1, X2, walk):-
        derive1([animal], [], X1, X2, _).
mode(X1, X2, run):-
        derive1([animal], [], X1, X2, Y),
        no_of_legs(Y, [], X),
        X >= 2.
mode(X1, X2, gallop):-
        derive1([animal], [], X1, X2, Y),
        no_of_legs(Y, [], X),
        X >= 4.

% class horse.

no_of_legs(X1, X2, 4):-
        derive1([horse], [], X1, X2, _).
covering(X1, X2, hair):-
        derive1([horse], [], X1, X2, _).

% class bird.

no_of_legs(X1, X2, 2):-
        derive1([bird], [], X1, X2, _).
covering(X1, X2, feather):-
        derive1([bird], [], X1, X2, _).
mode(X1, X2, fly):-
        derive1([bird], [], X1, X2, _).

% class tweety.

owner(X1, X2, fred):-
        derive1([tweety], [], X1, X2, _).

% class human(S, A).
```

```
sex(X1, X2, S):-
        derive1([human(S,A)], [],  X1, X2, _).
age(X1, X2, A):-
        derive1([human(S,A)], [], X1, X2, _).
no_of_legs(X1, X2, 2):-
        derive1([human(S,A)], [], X1, X2, _).
likes(X1, X2, logic):-
        derive1([human(S,A)], [], X1, X2, Y),
        sex(Y, [], male),
        age(Y, [], Ag),
        Ag < 40.
likes(X1, X2, logic):-
        derive1([human(S,A)], [], X1, X2, Y),
        sex(Y, [], female).

% class mathematician.

likes(X1, X2, logic):-
        derive1([mathematician], [], X1, X2, _).
likes(X1, X2, math):-
        derive1([mathematician], [], X1, X2, _).
```

Index of Symbols

Bibliography

ABADI, M. AND MANNA, Z. (1989). Temporal Logic Programming. *Journal of Symbolic Computation*, 8(3):277–295. [**xi, 83, 135**]

AKAMA, S. (1986). A Proposal of Modal Logic Programming. In *Proc. of the 6th Canadian Conference on Artificial Intelligence*, pages 99–102. [**xi, 134, 135**]

ARTOSI, A., BENASSI, P., GOVERNATORI, G., AND ROTOLO, A. (1996). Labelled Proofs for Quantified Modal Logic. In Alferes, J. J., Pereira, L. M., and Orlowska, E., editors, *Logics in Artificial Intelligence, JELIA '96*, volume 1126 of *LNAI*, pages 70–86. Springer-Verlag. [**72, 73**]

AUFFRAY, Y. AND ENJALBERT, P. (1992). Modal Theorem Proving: An equational viewpoint. *Journal of Logic and Computation*, 2(3):247–297. [**74, 101**]

BALBIANI, P., FARIÑAS DEL CERRO, L., AND HERZIG, A. (1988). Declarative semantics for modal logic programs. In *Proc. of the International Confonference on Fifth Generation Computer Systems, FGCS'88*, pages 507–514, Tokyo. [**xi, 123, 128, 134, 135**]

BALBIANI, P., HERZIG, A., AND LIMA MARQUES, M. (1991). TIM: The Toulouse inference machine for non-classical logic programming. In *Proc. of the International Workshop on Processing Declarative Knowledge, PDK'91*, volume 567 of *LNAI*, pages 366–382. Springer-Verlag. [**xi, 134**]

BALDONI, M., GIORDANO, L., AND MARTELLI, A. (1993). A Multimodal Logic to define Modules in Logic Programming. In Miller, D., editor, *Proc. of the International Logic Programming Symposium, ILPS'93*, pages 473–487, Vancouver. The MIT Press. [**77, 84, 88, 105, 131–134**]

BALDONI, M., GIORDANO, L., AND MARTELLI, A. (1996a). A Framework for Modal Logic Programming. In Maher, M., editor, *Proc. of the Joint International Conference and Symposium on Logic Programming, JICSLP'96*, pages 52–66, Bonn. The MIT Press. [**134**]

BALDONI, M., GIORDANO, L., AND MARTELLI, A. (1996b). Translating a Modal Language with Embedded Implications into Horn Clause Logic. In Dyckhoff, R., Herre, H., and Schroeder-Heister, P., editors, *Proc. of the 5th International Workshop on Extensions of Logic Programming, ELP'96*, volume 1050 of *LNAI*, pages 19–33. Springer-Verlag. [**97, 133**]

Baldoni, M., Giordano, L., and Martelli, A. (1997a). A Modal Extention of Logic Programming: Modularity, Beliefs and Hypothetical Reasoning. Technical Report RT 36/97, Dipartimento di Informatica, University of Turin. Accepted for the pubblication in the *Journal of Logic and Computation.* A short version appears in the Proc. of the *1994 Joint Conference on Declarative Programming, GULP-PRODE 1995*, pages 324–335. [**xi, 49, 77, 84, 96, 105, 132–134, 139**]

Baldoni, M., Giordano, L., and Martelli, A. (1998a). A Tableau Calculus for Multimodal Logics and Some (Un)Decidability Results. In de Swart, H., editor, *Proc. of the International Conference on Analytic Tableaux and Related Methods, TABLEAUX'98.* Springer-Verlag. To appear in the LNAI serie. [**xi**]

Baldoni, M., Giordano, L., Martelli, A., and Patti, V. (1997b). An Abductive Proof Procedure for Reasoning about Actions in Modal Logic Programming. In Dix, J., Pereira, L. M., and Przymusinski, T. C., editors, *Proc. of the 2nd International Workshop on Non-Monotonic Extensions of Logic Programming, NMELP'96*, volume 1216 of *LNAI*, pages 132–150. Springer-Verlag. [**xi, 104, 134, 138**]

Baldoni, M., Giordano, L., Martelli, A., and Patti, V. (1998b). A Modal Programming Language for Representing Complex Actions. Technical report, Dipartimento di Informatica, Univeristà degli Studi di Torino. [**138**]

Baudinet, M. (1989). Temporal logic programming is complete and expressive. In *Proc. of the 16th ACM Symposium on Principle of Programming Languages, POPL'89*, pages 267–280, Austin, Texas. [**xi, 123, 128**]

Beckert, B. and Goré, R. (1997). Free Variable Tableaux for Propositional Modal Logics. In Galmiche, D., editor, *Proc. of the International Conference on Automatic Reasoning with Analytic Tableaux and Related Methods, TABLEAUX'97*, volume 1227 of *LNAI*, pages 91–106. Springer-Verlag. [**xi, 7**]

Bondarenko, A., Toni, F., and Kowalski, R. A. (1993). An Assuption-based Framework for Non-monotonic Reasoning. In Pereira, L. M. and Nerode, A., editors, *Proc. of 2nd International Workshop on Logic Programming and Non-monotonic Rasoning*, pages 171–189. The MIT Press. [**134**]

Bonner, A. J., McCarty, L. T., and Vadaparty, K. (1989). Expressing database queries with intuitionistic logic. In Lusk, L. and Overbeek, R. A., editors, *Proc. of the 1989 North American Conference on Logic Programming*, pages 831–850. The MIT Press. [**119**]

Book, R. V. (1987). Thue Systems as Rewriting Systems. *Journal of Symbolic Computation*, 3(1-2):39–68. [**37, 78, 85**]

Bowen, K. A. and Kowalski, R. A. (1982). Amalgamating Language and Metalanguage in Logic Programming. In Clark, K. and Tarnlund, S., editors, *Logic Programming*, pages 153–172. Academic Press. [**105**]

Brogi, A., Lamma, E., and Mello, P. (1990a). A general framework for structuring logic programs. Technical Report 4/1, Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo — Sottoprogetto 4, Linguaggi di Nuova Concezione. [**78, 115–117**]

Brogi, A., Lamma, E., and Mello, P. (1990b). Inherritance and Hypothetical Reasoning in Logic Programming. In *Proc. of the European Conference on Artificial Intelligence, ECAI'90*, pages 105–110, Stockholm. [**78, 112, 114, 115, 117**]

Brogi, A., Mancarella, P., Pedreschi, D., and Turini, F. (1992). Meta for Modularising Logic Programming. In *Proc. of the META'92*, Stockholm. [**105**]

Brogi, A., Mancarella, P., Pedreschi, P., and Turini, F. (1994). Modular Logic Programming. *ACM Transactions on Programming Languages and Systems*, 16(4):1361–1398. [**105**]

Bugliesi, M. (1992). A Declarative View of Inheritance in Logic Programming. In Apt, K., editor, *Proc. of the Join International Conference and Symposium on Logic Programming, JICSLP'92*, pages 113–127, Washington. The MIT Press. [**107, 112**]

Bugliesi, M., Lamma, E., and Mello, P. (1994). Modularity in Logic Programming. *Journal of Logic Programming*, 19 & 20:443–502. [**xiii, 77, 105, 112**]

Catach, L. (1988). Normal Multimodal Logics. In *Proc. of the 7th National Conference on Artificial Intelligence, AAAI '88*, volume 2, pages 491–495, Sait Paul, Minnesota. Morgan Kaufmann. [**4, 7, 12, 53, 55, 57, 58, 69**]

Catach, L. (1991). TABLEAUX: A General Theorem Prover for Modal Logics. *Journal of Automated Reasoning*, 7(4):489–510. [**xi, 4, 22, 73**]

Chellas, B. F. (1980). *Modal Logic: an Introduction.* Cambridge University Press. [**7, 12, 42, 55, 57**]

Chen, C. C. and Lin, I. P. (1994). The computational complexity of the satisfiability of modal Horn clauses for modal propositional logics. *Theoretical Computer Science*, 129:95–121. [**139**]

Chen, W. (1987). A Theory of Modules based on Second Order Logic. In *Proc. of the Intenational Symposium on Logic Programming, ILPS'87*, pages 24–33, S. Francisco. [**105**]

Cialdea, M. and Fariñas del Cerro, L. (1986). A modal Herbrand's property. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 32(6):523–530. [**128**]

Cunningham, J. and Pitt, J. (1996). Distributed Modal Theorem Proving with KE. In Miglioli, P., Moscato, U., Mundici, D., and Ornaghi, M., editors, *Proc. of the 5th International Workshop on Theorem Proving with Analitic Tableaux and Related Methods, TABLEAUX '96*, volume 1071 of *LNAI*, pages 160–176. Springer-Verlag. [**xi, 7, 22, 72, 73**]

D'Agostino, M. and Modadori, M. (1994). The Taming of the Cut. *Journal of Logic and Computation*, 4:285–319. [**72**]

De Giacomo, G. and Lenzerini, M. (1995). PDL-based framework for reasoning about actions. In *Topics of Artificial Intelligence. Proc. of AI*IA'95, National Congress of the Italian Association for Artificial Intelligence*, volume 992 of *LNAI*, pages 103–114. Springer-Verlag. [**xi**]

De Giacomo, G. and Massacci, F. (1996). Tableaux and Algorithms for Propositional Dynamic Logic with Converse. In McCune, W., editor, *Automated Deduction — CADE-15*, volume 1249 of *LNAI*, pages 613–627. Springer. [**22, 69, 139**]

Debart, F., Enjalbert, P., and Lescot, M. (1992). Multimodal logic programming using equational and order-sorter logic. *Theoretical Computer Science*, 105(1):141–166. [**xi, 135**]

Dung, P. M. (1993a). Negations as Hypothesis: an Abductive Foundation for Logic Programming. In Furukawa, K., editor, *Proc. of International Conference on Logic Programming, ICLP'91*, pages 852–857. The MIT Press. [**134**]

Dung, P. M. (1993b). On the Acceptability of Arguments and its fundamental role for Logic Programming. In *Proc. of International Joint Conference on Articificial Intelligence, IJCAI'93*, pages 852–857. Morgan Kaufmann. [**134**]

Enjalbert, P. and Fariñas del Cerro, L. (1989). Modal Resolution in Clausal Form. *Theoretical Computer Science*, 65(1):1–33. [**xi, 12, 55**]

Fariñas del Cerro, L. (1986). MOLOG: A System that extends Prolog with Modal Logic. *New Generation Computing*, 4(1):35–50. [**xi, 134**]

Fariñas del Cerro, L. and Herzig, A. (1995). Modal Deduction with Applications in Epistemic and Temporal Logics. In Gabbay, D. M., Hogger, C., and Robinson, J., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 499–594. Oxford Science Publications. [**xi, 4, 12, 15**]

Fariñas del Cerro, L. and Penttonen, M. (1987). A note on the complexity of the satisfiability of modal Horn clauses. *Journal of Logic Programming*, 4:1–10. [**139**]

Fariñas del Cerro, L. and Penttonen, M. (1988). Grammar Logics. *Logique et Analyse*, 121-122:123–134. [**xii, 5, 6, 17, 18, 33, 34, 37**]

Fariñas del Cerro, L. and Penttonen, M., editors (1992). *Intensional Logics for Programming.* Studies in Logic and Computation. Oxford Science Pubblications. [**xi**]

Fischer, M. J. and Ladner, R. E. (1979). Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Sciences*, 18(2):194–211. [**7, 33, 39**]

Fisher, M. and Owens, R., editors (1993a). *Executable Modal and Temporal Logics*, volume 897 of *LNAI.* Springer-Verlag. [**xi**]

Fisher, M. and Owens, R. (1993b). An Introduction to Executable Modal and Temporal Logics. In *Proc. of the IJCAI'93 Workshop on Executable Modal and Temporal Logics*, volume 897 of *LNAI*, pages 1–20. Springer-Verlag. [**77**]

Fitting, M. (1973). Model Existence Theorems for Modal and Intuitionistic Logics. *Journal of Symbolic Logic*, 38(4):613–627. [**22, 30**]

Fitting, M. (1983). *Proof Methods for Modal and Intuitionistic Logics*, volume 169 of *Synthese library.* D. Reidel, Dordrecht, Holland. [**xi, xii, 6, 21, 22, 28, 29, 48, 49, 51, 69, 70, 96, 138**]

Fitting, M. (1988). First-Order Modal Tableaux. *Journal of Automated Reasoning*, 4:191–213. [**xi**]

Fitting, M. (1993). Basic Modal Logic. In Gabbay, D., Hogger, C. J., and Robinson, J. A., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 365–448. Oxford Science Publications. [**xi, xii, 49**]

Fitting, M. (1996). Introduction to the handbook of tableau methods. In D'Agostino et al., M., editor, *Handbook of Tableau Methods*, pages 1–47. Kluwer Academic Press. To appear. [**96, 139**]

Gabbay, D. M. (1985). NProlog: An Extension of Prolog with Hypothetical Implications. II. *Journal of Logic Programming*, 2(4):251–283. [**132, 133**]

Gabbay, D. M. (1987). Modal and Temporal Logic Programming. In Galton, A., editor, *Temporal Logics and Their Applications*, pages 197–237. Academic Press. [**xi, 135**]

Gabbay, D. M. and Reyle, N. (1984). NProlog: An Extension of Prolog with Hypothetical Implications. I. *Journal of Logic Programming*, 4:319–355. [**132, 133**]

Garson, J. W. (1984). Quantification in Modal Logic. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic*, pages 249–307. Reidel. [**48**]

Gasquet, O. (1993). Optimization of deduction for multi-modal logics. In Masuch, Marx, and Plòs, editors, *Applied Logic: How, What and Why?* Kluwer Academic Publishers. [**5, 74, 136**]

Gasquet, O. (1994). *Déduction automatique en logique multi-modale par traduction.* PhD thesis, Institute de recherche en informatique de Toulose, Université Paul Sabatier, Toulose III. [**5, 48**]

Gelfond, M. and Lifschiftz, V. (1993). Representing Action and Change by Logic Programming. *Journal of Logic Programming*, 17:301–321. [**134**]

Genesereth, M. and Nilsson, N. (1987). *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann. [**xi, 12, 13, 58, 104**]

Giordano, L. and Martelli, A. (1992). A modal framework for structured logic programs. In Lamma, E. and Mello, P., editors, *Proc. of Third Int. Workshop on Extensions of Logic Programs, ELP'96*, volume 660 of *LNAI*, pages 168–186, Bologna. Springer-Verlag. [**116, 118**]

Giordano, L. and Martelli, A. (1994). Structuring Logic Programs: a Modal Approach. *Journal of Logic Programming*, 21(2):59–94. [**xi, 77, 84, 105, 116, 132, 133, 139**]

Giordano, L., Martelli, A., and Rossi, G. F. (1992). Extending Horn Clause Logic with Implication Goals. *Theoretical Computer Science*, 95:43–74. [**105, 132**]

Giordano, L., Martelli, A., and Rossi, G. F. (1994). Structured Prolog: A language for Structured Logic Programming. *Software - Concept and Tools*, 15:125–145. [**110**]

Goldberg, A. and Robson, D. (1983). *Smalltalk-80 The Language and its Implementation.* Addison-Wesley Publishing Company. [**109**]

Goré, R. A. (1995). Tableaux Methods for Modal and Temporal Logics. Technical Report TR-ARP-16-95, Automated Reasoning Project, Australian National University. To appear in In M. D'Agostino et al., editor, *Handbook of Tableau Methods.* Kluwer Academic Press. [**22, 28, 30, 69, 70**]

Governatori, G. (1995). Labelled Tableaux for Multi-Modal Logics. In Baumgartner, P., Hähnle, R., and Posegga, J., editors, *Proc. of the 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, TABLEAUX '95*, volume 918 of *LNAI*, pages 79–94. Springer-Verlag. [**xi, 7, 22, 72, 73**]

Governatori, G. (1997). *Un modello formale per il ragionamento giuridico.* Phd. thesis, University of Bologna, Bologna. In Italian. [**72, 73**]

Halpern, J. Y. and Moses, Y. (1992). A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief. *Artificial Intelligence*, 54:319–379. [**xi, 3, 5, 7, 11–13, 16–18, 55, 58, 103, 104, 132**]

Harel, D. (1984). Dynamic Logic. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic*, volume II, pages 497–604. D. Reidel Publishing Company. [**33, 58**]

Harel, D. and Paterson, M. S. (1984). Undecidability of PDL with $L = \{a^{2^i} \mid i \geq 0\}$. *Journal of Computer and System Sciences*, 29:359–365. [**33**]

Harel, D., Pnueli, A., and Stavi, J. (1983). Propositional Dynamic Logic of Nonregular Programs. *Journal of Computer and System Sciences*, 26:222–243. [**33**]

Hill, P. M. (1993). A Parametrized Module System for Costructing Typed Logic Programs. In *Proc. of International Joint Conference on Artificial Intelligence, IJCAI'93*, volume 2, pages 874–880. Morgan Kaufmann. [**110**]

Hodas, J. and Miller, D. (1991). Logic Programming in a Fragment of Intuitionistic Linear Logic. In Kahn, G., editor, *Sixth Annual Symposium on Logic in Computer Science*, pages 32–42, Amsterdam. [**133**]

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to automata theory, languages, and computation.* Addison-Wesley Publishing Company. [**33, 34, 37, 38, 85, 101**]

Hughes, G. E. and Cresswell, M. J. (1968). *A Introduction to Modal Logic.* Methuen, London. [**xi, 48**]

Hughes, G. E. and Cresswell, M. J. (1984). *A Companion to Modal Logic.* Methuen. [**12, 40, 42, 55**]

Hughes, G. E. and Cresswell, M. J. (1996). *A New Introduction to Modal Logic.* Routledge. [**xi, 3, 7, 11, 12, 17, 18, 48, 55**]

Konolige, K. (1986). *A deduction model of belief.* Morgan Kaufmann Publishers. [**xi**]

Kozen, D. and Tiuryn, J. (1990). Logics of Programs. In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science*, volume B, pages 788–840. Elsevier Science Publishers. [**33, 58**]

Kracht, M. (1995). Highway to the Danger Zone. *Journal of Logic and Computation*, 5(1):93–109. [**37**]

Lamma, E., Mello, P., and Rossi, G. F. (1993). Parametric composable in a logic programming language. *Computer Languages*, 18(2):105–123. [**105, 107, 110, 115–117**]

Lloyd, J. W. (1984). *Foundations of Logic Programming.* Springer-Verlag. [**xi**]

Mancarella, P. and Pedreschi, D. (1988). An Algebra of Logic Programs. In *Proc. of the Fifth International Conference of Logic Programming, ICLP'88*, pages 1006–1023, Seattle. [**105**]

Massacci, F. (1994). Strongly Analytic Tableaux for Normal Modal Logics. In *Proc. of the CADE'94*, volume 814 of *LNAI*, pages 723–737. Springer-Verlag. [**xi, 22, 69, 70, 138**]

McCabe, F. G. (1992). *Logic and Objects.* International Series in Computer Science. Prentice-Hall. [**78, 107, 110, 112, 114**]

McCarty, L. T. (1988a). Clausal Intuitionistic Logic. I. Fixed-Point Semantics. *Journal of Logic Programming*, 5(1):1–31. [**132, 133**]

McCarty, L. T. (1988b). Clausal Intuitionistic Logic. II. Tableau Proof Procedure. *Journal of Logic Programming*, 5(2):93–132. [**132, 133**]

Miller, D. (1986). A Theory of Modules for Logic Programming. In *Proc. of the IEEE Symposium on Logic Programming*, pages 106–114. [**132, 133**]

Miller, D. (1989a). A Logical Analysis of Modules in Logic Programming. *Journal of Logic Programming*, 6(2):79–108. [**86, 105, 107, 115, 119, 133**]

Miller, D. (1989b). Lexical Scoping as Universal Quantification. In *Proc. of the 6th Interantional Conference on Logic Programming, ICLP'89*, pages 268–283, Lisbon. [**133**]

Miller, D., Nadathur, G., Pfenning, F., and Scedrov, A. (1991). Uniform Proofs as Foundations for Logic Programming. *Annals of Pure and Applied Logic*, 51:125–157. [**xiii, 78, 89, 94, 131–133**]

Monteiro, L. and Porto, A. (1989). Contextual Logic Programming. In Levi, G. and Martelli, M., editors, *Proc. of the 6th International Conference of Logic Programming, ICLP'89*, pages 284–299. The MIT Press. [**78, 105, 107, 108, 115**]

Monteiro, L. and Porto, A. (1990). A Transformational View of Inheritance in Logic Programming. In Warren, D. H. D. and Szeredi, P., editors, *Proc. of the 7th International Conference of Logic Programming, ICLP'90*, pages 481–494. The MIT Press. [**108, 110**]

Moore, R. C. (1980). *Reasoning about Knowledge and Action.* PhD thesis, MIT, Cambridge, Massachussets. [**73**]

Nait Abdallah, M. A. (1986). Ions and Local Definitions in Logic Programming. In *Proc. of the STACS'86*, volume 210 of *LNCS*, pages 60–72. Springer-Verlag. [**105**]

NERODE, A. (1989). Some Lectures on Modal Logic. In Bauer, F. L., editor, *Logic, Algebra, and Computation*, volume 79 of *NATO ASI Series*. Springer-Verlag. [**xii, 6, 21, 22**]

NONNENGART, A. (1993). First-Order Modal Logic Theorem Proving and Functional Simulation. In *Proc. of International Joint Conference on Artificial Intelligence, IJCAI'93*, pages 80–85. Morgan Kaufmann. [**74, 135**]

NONNENGART, A. (1994). How to use Modalities and Sorts in Prolog. In MacNish, C., Pearce, D., and Pereira, L. M., editors, *Proc. of the JELIA'94: Logics in Artificial Intelligence*, volume 838 of *LNAI*, pages 365–378, York, UK. Springer-Verlag. [**xi, 74, 135, 136**]

OGNJANOVIĆ, Z. (1994). A tableau-like proof procedure for normal modal logics. *Theoretical Computer Science*, 129:167–186. [**xi**]

OHLBACH, H. (1991). Semantics-Based Translation Methods for Modal Logics. *Journal of Logic and Computation*, 1(5):691–746. [**74, 101**]

OHLBACH, H. J. (1993a). Optimized Translation of Multi Modal Logic into Predicate Logic. In Voronkov, A., editor, *Proc. of the Logic Programming and Automated Reasoning*, volume 822 of *LNAI*, pages 253–264. Springer-Verlag. [**74**]

OHLBACH, H. J. (1993b). Translation methods for non-classical logics: An overview. *Bull. of the IGPL*, 1(1):69–89. [**73, 97, 101**]

O'KEEFE, R. A. (1985). Towards an Algebra for Constructing Logic Programs. In *Proc. of the Symposium on Logic Programming*, pages 152–160, Boston. [**105**]

ORGUN, M. AND MA, W. (1994). An overview of temporal and modal logic programming. In Gabbay, D. and Ohlbach, H., editors, *Proc. of the First International Conference on Temporal Logic*, volume 827 of *LNAI*, pages 445–479. Springer-Verlag. [**xi, 77**]

ORGUN, M. AND WADGE, W. W. (1992). Towards a unified theory of intensional logic programming. *Journal of Logic Programming*, 13(4):413–440. [**123, 128**]

SAKAKIBARA, Y. (1986). Programming in Modal Logic: An extension of PROLOG based on Modal Logic. In Wada, E., editor, *Logic Programming '86*, volume 264 of *LNCS*, pages 81–91. Springer-Verlag. [**xi**]

SMULLYAN, R. M. (1968). *First-Order Logic*, volume 43 of *Ergebnisse der Mathematik*. Springer-Verlag, Berlin. [**22**]

STIRLING, C. (1992). Modal and temporal logics. In Abramsky, S., Gabbay, D. M., and Maibaum, T. S. E., editors, *Handbook of Logic in Computer Science*, volume 2, pages 477–563. Clarendon Press, Oxford. [**xi**]

TURINI, F. (1995). Extensions of Logic Programming in support of Software Engineering. In Sessa, M. I., editor, *1985-1995 Ten years of Logic Programming in Italy*, pages 241–272. Palladio. [**78, 112**]

WALLEN, L. A. (1990). *Automated Deduction in Nonclassical Logics.* The MIT Press. [**xi, 49, 96**]

WOOLDRIDGE, M. AND JENNINGS, N. R. (1995). Agent Theories, Architectures, and Languages: A survey. In *Proc. of the ECAI-94 Workshop on Agent Theories*, volume 890 of *LNAI*, pages 1–39. Springer-Verlag. [**xi**]