HAL open science

Modal memory logics

Sergio Fernando Mera

▶ **To cite this version:**

Sergio Fernando Mera. Modal memory logics. Other [cs.OH]. Université Henri Poincaré - Nancy 1, 2009. English. NNT : 2009NAN10130 . tel-01748219

HAL Id: tel-01748219

https://hal.univ-lorraine.fr/tel-01748219

Submitted on 29 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE LORRAINE

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

UFR STMIA - École doctorale IAEM Lorraine Département de formation doctorale en informatique

Modal memory logics
Logiques modales mémorielles

Thèse présentée et soutenue publiquement le decembre 2009
pour l'obtention du

Doctorat de l'université Henri Poincaré, Nancy 1
(spécialité Informatique)

par

Sergio Fernando Mera

Composition du jury

Rapporteurs:	Stéphane Demri	Directeur de recherche, CNRS
	Sergio Yovine	Directeur de recherche, CNRS
Examinateurs:	Alfredo Olivero	Professeur, UADE
	Carlos Areces	INRIA Nancy - Grand Est
	Patrick Blackburn	INRIA Nancy - Grand Est
	Verónica Becher	Universidad de Buenos Aires

Laboratoire Lorrain de Recherche en Informatique et ses Applications – UMR 7503

Contents

Acknowledgments **vii**

Abstract **ix**

Resumen **xi**

1 A zoo full of logics **1**
1.1 The place of Modal Logics 2
1.1.1 A little of modal history 4
1.1.2 A contemporary perspective 5
1.2 A formal introduction 6
1.2.1 Some modal examples 9
1.3 A wide menu to choose from 14
1.3.1 The universal modality 14
1.3.2 Since and Until operators 15
1.3.3 Hybrid Logics 16
1.4 Modifying models: memory and state 19
1.5 Overview of the thesis 21

2 Memory Logics **23**
2.1 Focusing on sets 23
2.1.1 Extended operators 25
2.1.2 Memorizing policies 27
2.1.3 Putting it all together 28
2.2 Getting to know your logic 30
2.2.1 Expressivity 31
2.2.2 Decidability 36
2.2.3 Interpolation and Beth definability 39
2.2.4 Axiomatizations 40

2.2.5 Tableau calculus 42
2.3 How memory logics were born? 45

3 Expressiveness **47**
3.1 Model equivalence 47
3.2 Expressive power 53
3.2.1 Logics with an initially empty memory 54
3.2.2 Erase and Forget operators 58
3.2.3 Memory logics with a stack 62
3.2.4 Comparing apples with oranges 64
3.3 A general picture 65

4 (Un)Decidability **67**
4.1 The decidability of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ 67
4.1.1 A linear encoding for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ 71
4.2 Some undecidable neighbors of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ 72
4.2.1 Just one nominal is enough 72
4.2.2 An empty memory is also enough 75

5 Interpolation and Beth definability **79**
5.1 Interpolation fails 80
5.2 Some positive results 83
5.2.1 Preliminaries 83
5.2.2 The main result 85
5.3 The quest for Beth definability 90
5.3.1 Some negative results 92
5.4 Observations 94

6 Axiomatizations **95**
6.1 Completeness for $\mathcal{HL}^m(@, \langle r\rangle)$ 96
6.1.1 Pure extensions 100
6.2 Dealing with other memory operators 101
6.2.1 Adding the $\textcircled{e}$ operator 102
6.2.2 Adding the $\textcircled{f}$ operator 103
6.2.3 The operators $\textcircled{e}$ and $\textcircled{f}$ together 106
6.3 The case for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ 107
6.3.1 An extension to $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ 111
6.4 Some final remarks 113

7 Tableau systems **115**
7.1 A tableau system for $\mathcal{ML}^m(\langle r\rangle, \textcircled{e}, \textcircled{f})$ 116
7.2 A terminating tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ 121
7.2.1 Showing termination 124

7.3 Tableau or axiomatic systems? 126

8 Conclusions **129**
8.1 A review of what we did 130
8.2 How do we continue? 133

Bibliography **135**

Acknowledgments

My PhD studies formally started in 2005, but the first ideas about memory logics and the work that led to this thesis began in mid 2006. I have been supported and helped by many people throughout these five years, and I would like to express my gratitude to all of them.

First of all, I would like to give my deep and sincere thanks to my supervisor, Carlos Areces. Thanks for sharing many, many things with me, not all of them related with Logic. Thanks for the trips we made, the music, movies and books, for having been welcomed into his home many times, and for being such a great cook (if the reader has not yet tried Carlos' octopus recipe, I strongly suggest doing so ⌣̈). Thanks for reading so carefully all the things we wrote together, and for teaching me the importance of telling a consistent story, beyond the technical results. I am also indebted to him for giving me the chance to meet many interesting people at several meetings, and the possibility to get to know the people from LORIA, with whom I shared lots of great moments. I am also grateful for all the work Carlos did to establish the Group in Logic and Computability (GLyC), and all the effort he put in making it grow.

I also want to express my gratitude to the people who worked with me in many of the results that are in this thesis: Santiago Figueira and Daniel Gorín, with whom I worked side by side, Mariano Pérez Rodríguez, who came with the idea that was the origin of memory logics, and Diego Figueira, for his enormous help to find the countermodels shown in Chapter 3 (also known as "*la demoledora*" and "*la doble Nelson*"). I am also grateful to the other logicians of GLyC, Ricardo Rodríguez, Facundo Carreiro and Daniel Koile, for the many enriching discussions we had and for the interest in my work.

I am particularly grateful to Patrick Blackburn and Verónica Becher for making this cotutelle possible. I want to especially thank Verónica for introducing me to Carlos, and for suggesting that a PhD under Carlos' supervision was a good idea (which indeed it was!). I also want to thank Patrick for all the valuable discussions we had at LORIA, and for being always ready to read any scribble I

had, no matter how unpolished it was.

The people from Talaris were really important to create the extremely nice atmosphere I always found during all my stays in France. I want to thank Lina María Rojas Barahona (especially for disagreeing with my "and but" style), Luciana Benotti, Alejandra Lorenzo, Laura Perez, Corinna Anderson, Humberto Abdelnur, Yannick Parmentier, Sébastien Hinderer, Luis Peñaranda and the wooden table at LORIA. A special thanks goes to Guillaume Hoffman, that translated the abstract to French. In the same way, I am grateful to all the people of the Department of Computer Science at the University of Buenos Aires, for providing an unbeatable working environment. I would like to be fair, and mention each of them, as I did with the people from Talaris. But they are so many that I'm sure that I will forget lots of names, and I would not like that to happen.

Finally, I would like to thank my family, for all the support they gave me, and for always encouraging me in every decision I made.

Abstract

From ancient times to the present day, the field of logic has gained significant strength and now it actively contributes to many different areas, such as philosophy, mathematics, linguistic, computer science, artificial intelligence, hardware manufacture, etc. Each of these scenarios has specific needs, that range from very concrete requirements, like an efficient inference method, to more abstract theoretical properties, like a neat axiomatic system. Given this wide diversity of uses, a motley collection of formal languages has been developed. For many years, classical languages (mainly classical first order logic) were *the* alternative, but this assortment of applications made other types of logics also attractive in many situations.

Imagine that the time for choosing a logic for some specific task arrives. How can we decide which is the one that fits best? Which properties should we look for? How can we "measure" a logic with respect to others? These are not easy questions, and there is not a general recipe one can follow. In this thesis we are just going to restrict these questions to a particular family of logics, and in that context we will investigate theoretical aspects that help to answer some of these concerns. Much can be discovered by carefully analyzing appealing cases, and our contribution will be developed having that philosophy in mind.

Propositional modal logics offer an alternative to traditional languages. They can be regarded as a set of tools that allow to design logics specially tailored for specific tasks, having a fine-grained control on their expressivity. Additionally, modal logics turned out to have a good computational behavior, which proved to be quite robust under extensions. These characteristics, among others, placed modal logics as an attractive alternative to classical languages.

In this dissertation we are going to present a new family of modal logics called *memory logics*. Traditional modal logics enables to *describe* relational structures from a local perspective. But what about *changing* the structure? We want to explore the addition of an explicit storage structure to modal logics, a memory, that allows to model dynamic behavior through explicit memory operators.

These operators *store* or *retrieve* information to and from the memory. Naturally, depending on which type of storage structure we want, and which memory operators are available, the resulting logic will enjoy different properties that are worth investigating.

The thesis is organized as follows. In Chapter 1 we start by giving a brief recap of how modal logic was born, showing the different historical perspectives used to look at modal logic. Then we formally present the basic modal logic and a set of extended operators that helps grasp the modal "flavor" of some richer languages. We finish this chapter by giving a first glance of memory logics, and showing how they can help to model state when we choose to use a set as storage structure. Chapter 2 is devoted to present memory logics in detail. We show some examples that can be described by adding a set to standard relational structures, and the usual set operators to add elements and test membership. We then show some other memory operators that can be considered, and we discuss the possibility of adding constraints to the interplay between memory and modal operators. These constraints can be regarded as a way to have a finer-grained control on the logic expressivity. Since we have made changes to classical modal logics, we are interested in analyzing the impact those changes cause in the resulting logics. Therefore, the rest of this chapter presents a basic logic toolkit through which we can analyze this new family of logics. This toolkit can be seen as an outline that organizes the rest of the thesis and that allows to analyze memory logics in terms of expressivity, complexity, interpolation and proof theory.

The rest of the chapters investigate each of these aspects in detail. In Chapters 3 and 4 we explore the expressive power of several memory logics and we study the decidability of their satisfiability problem. In the decidable cases, we determine their computational complexity. We analyze the impact of the different memory operators we consider, and how they interact. We also study other memory containers, such as a stack. Then, in Chapter 5, we analyze Craig interpolation and Beth definability for some memory logic fragments. We also study memory logics from a proof theoretical perspective. In Chapter 6 and 7 we turn to Hilbert style axiomatizations and tableau systems, and we characterize several fragments of the memory logic family mostly using techniques borrowed from hybrid logics. We close in Chapter 8 with some concluding remarks, open problems and directions for further research.

Resumen

Desde la antigüedad hasta hoy, el campo de la lógica ha ido ganando fuerza y actualmente constribuye activamente en muchas áreas distintas, como filosofía, matemática, lingüística, ciencias de la computación, inteligencia artificial, fabricación de hardware, etc. Cada uno de estos escenarios tiene necesidades específicas, que van desde requerimientos muy concretos, como un método de inferencia eficiente, hasta propiedades teóricas más abstractas, como un sistema axiomático elegante. Durante muchos años, los lenguajes clásicos (principalmente la lógica de primer orden) eran *la* alternativa a utilizar, pero esta gran variedad de aplicaciones hizo que otro tipo de lógicas empezaran a resultar atractivas en muchas situaciones.

Supongamos que llega el momento de elegir una lógica para una tarea en particular. ¿Cómo podemos decidir cuál es la más adecuada? ¿Qué propiedades deberíamos buscar? ¿Cómo podemos "medir" una lógica con respecto a otras? Éstas no son preguntas sencillas, y no hay una receta general que uno pueda seguir. En esta tesis vamos a restringir estas cuestiones a una familia de lógicas en particular, y en ese contexto vamos a investigar algunos aspectos teóricos que nos van a ayudar a responder parte de estas inquietudes. Podemos aprender mucho estudiando casos particularmente interesantes, y nuestra contribución se va a desarrollar teniendo esa filosofía en mente.

Las lógicas modales proposicionales ofrecen una alternativa a los lenguajes tradicionales. Pueden ser pensadas como un conjunto de herramientas que permiten diseñar lógicas específicamente construidas para una tarea en particular, posibilitando tener un control fino en su expresividad. Más aún, las lógicas modales resultaron tener un buen comportamiento computacional que probó ser bastante robusto frente a extensiones. Estas características, entre otras, ubicaron a las lógicas modales como una alternativa atractiva con respecto a los lenguajes clásicos.

En este trabajo vamos a presentar una nueva familia de lógicas modales llamada *memory logics*. Las lógicas modales tradicionales posibilitan *describir* es-

tructuras relacionales desde una perspectiva local, ¿pero cuál será el resultado si permitimos que una fórmula *cambie* la estructura en la que está siendo evaluada? Queremos explorar el efecto de agregar a las lógicas modales clásicas una estructura de almacenamiento explícita, una *memoria*, que permita modelar comportamiento dinámico a través de operadores que permitan *almacenar* y *recuperar* información de la memoria. Naturalmente, dependiendo del tipo de estructura de almacenamiento que utilicemos, y de qué operadores tengamos disponibles, la lógica resultante va a tener distintas propiedades que valen la pena investigar.

Esta tesis está organizada de la siguiente manera. En el Capítulo 1 empezamos dando un breve resumen de cómo nacieron las lógicas modales, mostrando las diferentes perspectivas con las que históricamente se miró a la lógica modal. Luego presentamos formalmente a la lógica modal básica, y a un conjunto extendido de operadores que permiten apreciar el "estilo" modal de algunos lenguajes más ricos. Este capítulo finaliza con un primer bosquejo de las *memory logics*, mostrando cómo pueden ayudar a modelar la noción de estado cuando fijamos a un conjunto como estructura de almacenamiento. El capítulo 2 está dedicado a presentar a las *memory logics* con detalle. En dicho capítulo mostramos algunos ejemplos que pueden ser descriptos agregando un conjunto a las estructuras relacionales estándar, junto con los operadores usuales sobre conjuntos que permiten agregar elementos y verificar pertenencia. A continuación mostramos otros operadores sobre conjuntos que pueden ser considerados, y discutimos la posibilidad de agregar restricciones a la interacción entre los operadores que trabajan sobre la memoria y los operadores modales. Estas restricciones pueden ser pensadas como una manera de lograr un control más fino en la expresividad. Dado que realizamos cambios en las lógicas modales clásicas, estamos interesados en analizar el impacto que esos cambios causaron en las lógicas resultantes. Por lo tanto, el resto del capítulo presenta un conjunto de herramientas a través de las cuales analizamos esta nueva familia de lógicas. Este conjunto de herramientas puede verse como un esquema que organiza el resto de la tesis, y que permite analizar a las *memory logics* en términos de expresividad, complejidad, interpolación y teoría de prueba.

El resto de los capítulos investigan cada uno de estos aspectos en detalle. En los Capítulos 3 y 4 exploramos el poder expresivo de varias *memory logics* y estudiamos para cada fragmento la decidibilidad del problema de determinar la satisfactibilidad de una fórmula. En los casos decidibles, determinamos la complejidad computacional de cada uno. Analizamos el impacto de los diferentes operadores, su interacción, y también estudiamos otras estructuras de almacenamiento, como la pila. Luego, en el Capítulo 5, analizamos las propiedades de interpolación de Craig y definibilidad de Beth para algunas *memory logics*. También nos interesa estudiar a las *memory logics* desde la perspectiva de la teoría de prueba. En los Capítulos 6 y 7 nos volcamos al estudio de axiomatizaciones *à la* Hilbert y sistemas de tableau, y caracterizamos varios fragmentos usando principalmente técnicas utilizadas en lógicas híbridas. En el Capítulo 8 presentamos nuestras

conclusiones, mencionamos algunos problemas abiertos y futuras direcciones de investigación.

Résumé

Depuis l'antiquité jusqu'à aujourd'hui, le domaine de la logique a gagné une importance remarquable et contribue désormais à de nombreuses autres branches, telles que la philosophie, les mathématiques, la fabrication de matériel informatique, la linguistique, l'informatique, l'intelligence artificielle, etc. À chacun de ces scénarios correspondent des besoins spécifiques, qui vont d'exigences très concrètes, telles qu'une méthode d'inférence efficace, à des propriétés théoriques plus abstraites, telles qu'un système d'axiomes élégant. Étant donnée cette grande diversité d'utilisations, une palette hétéroclite de langages formels a été développée. Pendant de nombreuses années, les langages classiques (notamment la logique du premier ordre) étaient la *seule* alternative concevable, mais cet assortiment d'applications a rendu d'autres types de logiques également désirables dans de nombreuses situations.

Imaginez que l'heure de choisir une logique pour une tâche spécifique arrive. Comment choisir la plus appropriée? Quelles propriétés devrions-nous rechercher? Comment "mesurer" une logique par rapport aux autres? Ce sont des questions difficiles, et il n'existe pas de recette générale à suivre. Dans cette thèse, nous allons simplement restreindre ces questions à une famille particulière de logiques, et dans ce contexte, nous explorerons les aspects théoriques qui aideront à répondre à ces préoccupations. Beaucoup peut être découvert par une analyse attentive des cas les plus intéressants, et notre contribution sera développée selon cette philosophie.

Les logiques modales propositionnelles offrent une alternative aux langages traditionnels. Elles peuvent être considérées comme un ensemble d'outils permettant de concevoir des logiques adaptées à des tâches précises, possédant un contrôle fin sur leur expressivité. De plus, il s'est avéré que les logiques modales possèdent un bon comportement computationnel, qui se trouve être robuste y compris malgré l'ajout d'extensions. Ces caractéristiques, parmi d'autres, ont élevé les logiques modales au rang d'alternatives désirables aux langages classiques.

Dans ce thèse, nous allons présenter une nouvelle famille de logiques modales appelée *logiques mémorielles*. Les logiques modales traditionnelles permettent de *décrire* les structures relationnelles d'un point de vue local. Mais pourquoi ne pas *changer* cette structure? Nous voulons étudier l'ajout d'une structure de stockage explicite aux logiques modales, une mémoire, qui permet de modéliser un comportement dynamique à travers des opérateurs mémoriels explicites. Ces opérateurs *sauvent* ou *restaurent* de l'information vers et à partir de la mémoire. Naturellement, selon le type de structure de sauvegarde désiré et les opérateurs mémoriels disponibles, la logique résultante possèdera différentes propriétés qui valent la peine d'être étudiées.

Cette thèse est organisée de la façon suivante. Dans le Chapitre 1, nous commençons par rappeler brièvement comment la logique modale est née, en montrant les différents points de vue historiques la concernant. Puis, nous présentons formellement la logique modale de base et un ensemble d'opérateurs étendus qui aident à capturer le "goût" modal de langages plus riches. Nous finissons ce chapitre en donnant un premier aperçu des logiques mémorielles, et montrons comment elles peuvent aider à modéliser l'état quand nous choisissons d'utiliser un ensemble comme une structure de sauvegarde. Le Chapitre 2 est dédié à la présentation détaillée des logiques mémorielles. Nous montrons quelques exemples qui peuvent être décrits en ajoutant un ensemble à des structures relationnelles usuelles, ainsi que les opérateurs ensemblistes usuels permettant l'ajout d'élément et le test d'appartenance. Puis, nous montrons que d'autres opérateurs mémoriels peuvent être envisagés, et nous discutons de la possibilité d'ajouter des contraintes à l'interaction entre la mémoire et les opérateurs modaux. Ces contraintes peuvent être vues comme une manière d'avoir un contrôle fin sur l'expressivité de la logique. Comme nous avons fait des changements aux logiques modales classiques, nous nous intéressons à l'analyse de l'impact de ces changements sur les logiques résultantes. Ainsi, le reste de ce chapitre présente une boite à outils logique basique avec laquelle nous pouvons analyser cette nouvelle famille de logiques. Cette boite à outils peut être vue comme un plan qui organise le reste de cette thèse et qui permet d'analyser les logiques mémorielles en termes d'expressivité, de complexité, d'interpolation et de théorie de la preuve.

Le reste des chapitres consiste à étudier en détail chacun de ces aspects. Dans les Chapitres 3 et 4, nous explorons l'expressivité de plusieurs logiques mémorielles et nous étudions la décidabilité de leur problème de satisfiabilité. Dans les cas décidables, nous déterminons leur complexité. Nous analysons l'impact des différents opérateurs mémoriels considérés, et leur interaction. Nous étudions également d'autres conteneurs mémoriels, tels que la pile. Puis, dans le Chapitre 5, nous analysons l'interpolation de Craig et la définabilité de Beth pour certains fragments des logiques mémorielles. Nous étudions également les logiques mémorielles du point de vue de la théorie de la preuve. Dans les Chapitres 6 et 7, nous passons aux axiomatisations à la Hilbert et aux systèmes de tableaux, et nous caractérisons plusieurs fragments de la famille des logiques mémorielles, en

utilisant principalement des techniques empruntées aux logiques hybrides. Nous concluons dans le Chapitre 8 avec quelques remarques, des problèmes ouverts et des directions pour de futures recherches.

Chapter 1

A zoo full of logics

Sientes el vértigo que te captura la mirada y que se transforma en una comezón que desciende por tu espalda y alcanza tus manos que ahora se abren y se cierran por sí mismas sobre el hierro del parapeto: ahora sabes por qué Tadeus te ha atraído hasta allí, sólo él podía darte una cita como ésta.

"El ángel negro", Antonio Tabucchi.

The evolution of the logic research field seems to obey similar rules as natural evolution, in which constant change, diversity and specialization are some of the outstanding traits. This is not a surprise, since logic is a very active and evolving area of research, and each constituent of the logic family has to reassert its membership quite often, whenever new logics come into existence asking for more room. There are of course many perspectives from which logics can be analyzed, and as in many other flourishing scientific quests, to give an official view of which aspects would be desirable for a logic to have is often hard. Furthermore, the outcome of that definition would probably be unnecessarily strict. The analogy of thinking of logics as living and evolving entities can be helpful here, and the immediate question that it gives rise to is: when is a logic better than other? If we ask this to any logician, we would probably get new questions as answers: in which context are you going to use these logics? Which are the fundamental characteristics that you are looking for? In short, they will be asking: what is the meaning of 'better' for you?

Why can't we ask for all the pleasant features together? After all, we would like to have a nice, good and cheap logic. However, as one can naturally expect, logics behave as many other things in this universe, and it is not possible to have *the* perfect logic. In order to design a suitable logic, in which no valuable characteristic is neglected, a delicate balance is often needed, and this is the main cause of the existence of many different kinds of logics. When we have to choose a logic, we need to think first what is a valuable characteristic for us. This makes the previous questions boil down to: what is the appropriate logic that fits my specific problem?

The scenario in which a logician works varies enormously, since logic research was born in philosophy, but now it has spread into mathematics, linguistic, computer science, artificial intelligence, hardware manufacture and many other fields. This means that we are going to find a very crowded zoo out there, in which logics with many different colors, sizes and abilities happily live. The capabilities

that allow them to survive in their environments are also varied. Some logics were designed for very practical reasons, as they are able to express just the necessary things for the context they were designed for, and they admit relatively cheap computational procedures to perform reasoning tasks. These kind of logics are often used in areas such as formal software verification, digital hardware and programming language design, database management, logic programming, knowledge representation and model checking. In a nutshell, they live in the 'applied' side. At the opposite end of the spectrum, some other logics are valuable just because they have some very nice theoretical properties, and nobody will complain about their lack of applications.

A good example that illustrates the existence of different perspectives when evaluating a logic is the case of first order logic. First order logic offers high expressive power, a simple syntax and a neat model theory, but, on the other hand, its satisfiability problem is undecidable. If we think of an application that requires *feasible* inference, then perhaps first order logic is not the right choice. Since, most of the times, expressiveness and computational complexity are in the opposite sides of the scales, in many practical application first order logic is *unnecessarily expressive* at the high cost of undecidability. This is an important reason why first order logic has recently become less attractive as a representation formalism for many applications. In contrast, if we ask some purely theoretical logician his opinion about this matter, he could answer that he finds first order logic beautiful because it has a really nice axiomatic system. If someone points out to him the undecidability issue, he may even say that he finds undecidability *as a part* of the beauty of first order logic!

There are many good properties that a logic may have, such as decidability, appropriate expressive power, feasible inference methods, nice meta-logical properties (completeness, interpolation, etc.) and a language suitable for the context of use. As we said before, these properties do not always coexist peacefully together, and logicians put much effort in constructing new formalisms that try to balance some of these characteristics given a particular set of needs. In this context, the family of modal logics offers a wide menu to choose from that makes it a good framework for combining features.

1.1 The place of Modal Logics

The word 'modal' applied to logic has been inherited from the first attempts to formally study *modes of truth* and to try to pin down laws of reasoning for these modes. Let's try to intuitively grasp this idea. In classical logic (propositional or first order logic) formulas are either true or false in a model, but in many applications it is useful to distinguish between various 'modes' of truth. For example, the sentence

π is an irrational number

is currently true, but we can also try to think if there is a situation in which this claim is not true. This, *a priori*, seems difficult to imagine, since the sentence is formulating a mathematical principle. For example, the claim "there are nine planets in the solar system" is currently true, but clearly it is not true in every *possible* world, since we could had a solar system with more planets in some alternative world. On the other hand, the sentence "π is an irrational number", is *necessarily* true, since there is not a *possible* world in which is false. Other reasonable analysis is to think what happens with this claim in terms of the flow of time. The sentence "it is going to rain tomorrow" will be true at some point in the future, but definitively it is not true at every point. In contrast, "π is an irrational number" will *always be* true, since the truth of a mathematical principle does not seem to be affected by the flow of time. We could also be interested on whether the fact that π is an irrational number is known or not by somebody. Observe that this sentence is not *known* by everybody, since it implies some affinity with mathematical concepts. We can also think in terms of *belief*, and conclude that "π is an irrational number" is not universally *believed*, since there are people who do not trust in mathematicians.

That is, we can distinguish between various modes of truth, such as possibly true, believed to be true, known to be true, true at some point in time, etc. The first formal studies on modalities go back to the beginning of the XX century, with the work of C. I. Lewis [Lew18], who tried to solve the paradoxes of material implication using necessity and possibility. In fact there were other people before Lewis who built logic systems for this purpose, but Lewis seems to be the strongest link with contemporary modal logic. The problem that Lewis was trying to solve is that the natural language construction "if *condition* then *consequence*" implies some reasonable connection between the condition and the consequence. So, although the classical logic implication says that from a contradictory antecedent we always end up with a true claim, a phrase like "if the Earth has four moons, then the cube of 3 is 27" may be interpreted as false by most speakers, since the number of moons and the result of a mathematical operation are considered unrelated. Translated to a modern notation, Lewis idea was to take some assertion φ, and prefix it with a $\Box$ or a $\Diamond$ symbol, so $\Box\varphi$ says that the proposition φ is necessary and $\Diamond\varphi$ that the proposition φ is possible. Using modalities, Lewis sought to tighten up the connection between antecedent and consequent, and he introduces the *strict implication*, that can be defined as $\Box(p \to q)$. So p strictly implies q when the classical conditional *necessarily* holds in all possible worlds.

With this brief introduction we made, it is quite clear that modal logics actively contributes to keep the zoo of logics quite full. Modalities show to be a source of many different kinds of logics, and they expand the logics menu with new perspectives to consider.

1.1.1 A little of modal history

The work of Lewis gave rise to what some people call the 'syntatic era' of modal logics (starting in 1918), in which most of the work was essentially syntactic. Lewis and his contemporaries tried to determine rules of reasoning for the different ways of interpreting modalities, basically working with axiomatic systems *à la* Hilbert for propositional and first order logic enriched with some new modality. For example, if we want the intended interpretation of $\Box\varphi$ to be 'there is a proof for φ', then we would probably want to have $\Box\varphi \to \varphi$ (if there is a proof for φ, then φ must be true) as a theorem in our system. At that point there was no model theory for modal logics, and this entailed many technical difficulties. How can we be sure that two different axiomatizations model the same concept? Or how do we know that we have considered all the relevant axioms and rules given the intended interpretation of the modalities? Some examples of this kind of work can be found in [vW51, McK45, MT48] and for a detailed discussion of this period see the historical section of [BS84].

In the late 1950s and early 1960s, mainly pushed by the influence of Prior in temporal logic (or as he called it, *tense logic*) [Pri57] and Jónsson and Tarski with their work in the theory of boolean algebras with operators [JT51, JT52], semantics landed in the area of modal logics. In particular, Kripke's ideas on relational semantics turned up in this period. The actual authorship of relational semantics is in discussion (Hintikka, Kanger and Kripke are the best candidates), but relational semantics is often called Kripke semantics and definitively Kripke's work [Kri63a, Kri63b] was crucial in establishing the relational approach. Roughly speaking, Kripke's proposal was that a suitable model to evaluate a modal formula is just a set of possible worlds, with relations among them. The surprising thing when Kripke came with these ideas is that the different intended interpretations (like necessity, belief, knowledge, etc.), who seemed quite unrelated in terms of axiomatic systems, can be all characterized imposing structural properties in the relation among the worlds.

We already saw that necessity can be seen as "true at all possible worlds". Looking from the perspective of Kripke semantics, the current world is just some distinguished point w in a graph, and the outgoing edges of w represent all the "possible worlds" that are considered from the current world. The claim that says "it is necessary the case that π is an irrational number" can be translated to the formula $\Box$*pi-irrational-number*, where *pi-irrational-number* is a propositional symbol representing the fact that π is irrational. If we want to know whether this formula is true in the current world, we just have to check if at all the accessible successors of w the proposition *pi-irrational-number* holds.

Providing modal logics with a model theory was a revolutionary achievement. Many problems that had been thought of as difficult turned out to be quite straightforward using semantic arguments. Naturally, modal research at this point shift to semantically driven results, where the concept of *completeness* was

the leading figure: finally modal logicians could be sure that an axiomatic system has all the relevant axioms and rules needed to characterize the intended interpretation of a modality. The concept of *canonical models* emerged, a useful tool to prove completeness results, and they were applied to many problems that the syntactic era had left open. The work of Makinson [Mak66], Cresswell [Cre67] and Lemmon and Scott [LS77] are good representatives of this line of research.

At the beginning of 1970 several changes impacted the conception of what modal languages actually are. On one hand, in those years modal languages began to be adopted by theoretical computer science. For example, in [Pnu77] Pnueli suggested using temporal modal logic to reason about execution traces of programs, and the first computational complexity analysis (for example, Ladner [Lad77], Pratt [Pra79]) were brought to modal logics. On the other hand, the discovery of *frame incompleteness* results showed that there are classes of models for which there is no possible axiomatization (Thomason [Tho72, Tho74] and Fine [Fin74]). This shows that modal logics cannot be analyzed from a pure syntactic perspective. Also, solid links were established between modal logics and other classical logical systems (like first and second order logic, universal algebra and classical model theory, see [Sah73, Fin75, vB85, vB84]). These results helped to shift the view of modal logics as 'intensional' formalisms that were only able to talk about 'modes of truth' to a much broader panorama, which constitutes the current way of looking at modal logics.

1.1.2 A contemporary perspective

Nowadays modal logics can be thought of as a family of languages for talking about structures or models. What kind of structures? There is not a single kind, and although relational semantics is one of the most developed, there are many active lines of research in alternative semantics for modal logic (like algebraic or topological semantics, see [MT44, Esa74, She83, Tar38]). Relational semantics are a deeply explored style of modal semantics, and it has been used as a tool to talk about belief, computational processes, time, provability, possibility, etc [BdRV01, BWvB06]. All these very different areas have in common that the fundamental concepts they need to model can be expressed in terms of graphs-like structures. As one can notice, this definition is very inclusive, and a broad set of entities can be though of as relational structures: labeled transition systems, knowledge representation ontologies, tree structures used in linguistics, many familiar mathematical structures, etc. Furthermore, from a syntactic perspective, *propositional* modal languages, that is, the result of taking the well-known propositional logic as the underlying language and augment it with modal operators, are the simplest languages to deal with relational structures. This explains why there are so many potential (and concrete!) applications in which modal logics can be used to describe, reason about, verify and constrain the concepts involved in each specific scenario.

Modal logics have also proved to be a family of logics that is not isolated from the rest of the logical world. The semantics of modal logics can be thought of as a closed relative of the semantics of classical first order logic, but in which *explicit quantification* is restricted. The way to achieve this is by replacing classic quantification ($\forall$ and $\exists$) by *a priori* less expressive modal operators that do not make use of explicit variables and bindings.

Let's try to clarify this. Recall the example where $\Box\varphi$ represented 'necessarily φ' and $\Diamond\varphi$ 'possibly φ'. As we saw before, one way of interpreting 'necessarily φ' is to think that this claim holds when φ is true in *all* possible worlds. In the same way, 'possibly φ' is true when φ holds in *some* possible world. This interpretation suggests a strong link between the $\Box$ operator and the $\forall$ quantifier, and equivalently, between $\Diamond$ and $\exists$. The key difference lies in the fact that modal languages perform quantification in some kind of *guarded* way, in which the range of quantification is bounded to points that are considered significant from the point of view of the current 'situation'. In this sense, when modal languages are evaluated in a relational model they provide an *internal* perspective of its structure, from *a particular point*. As Blackburn, *et al.* say in [BdRV01],

> "a modal formula [can be seen as] a little automaton standing at some state in a relational structure, and only permitted to explore the structure by making journeys to neighboring states"

So modal languages inherit their semantics from the standard semantics of classical predicate logic, but with a 'tamed' way of doing quantification. On one hand, these similarities make modal logics have many properties in common with their classical counterpart. On the other hand, as modal operators usually have less expressive power than classical quantifiers, this results in many new properties, rather different from those of standard logic. Decidability is a clear example, since modal logics have helped discover many new decidable fragments of classical systems (even beyond first order logic!) whose computational complexity is often relatively low [Var97] (for example, the satisfiability problem for the basic modal logic is PSPACE-complete, see the complexity section in [BdRV01]). In this way, understanding modal logics as a fragment of first order, or even higher-order predicate logic does not bear any adverse connotation. This perspective shows modal logics as a set of fine-grained tools for exploring the inner structure of classical systems.

1.2 A formal introduction

It is now time to formally meet the modal logics we are going to work with and its relational semantics. We start by defining the *basic modal language* $\mathcal{ML}$. Because we are interested in working with many modalities at the same time, the diamond ($\Diamond$) and box ($\Box$) operators are going to turn into the operators $\langle r \rangle$ and

$[r]$, where r indicates the modality we are working with. When we are in a case where there is a single modality, we are going to use $\Diamond$ and $\Box$ again.

Although we introduce modal logic from scratch, we assume that the reader has at least a basic understanding of classical first order logic.

1.2.1. Definition. [Syntax] Suppose we have a set of propositional symbols $\textsc{prop} = \{p_1, p_2, \dots\}$ and a set of modality symbols $\textsc{rel} = \{r_1, r_2, \dots\}$. We assume that both sets are pairwise disjoint and countable infinite. A specific choice of prop and rel is called the *signature* of the language. We define the set of formulas of the *basic modal language* over the signature $\langle \textsc{prop}, \textsc{rel} \rangle$ as:

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \to \psi \mid \varphi \leftrightarrow \psi \mid \langle r \rangle \varphi \mid [r]\varphi$$

where $p \in \textsc{prop}$, $r \in \textsc{rel}$ and φ, ψ are formulas. We call $\mathsf{props}(\varphi)$ the set of propositional symbols occurring in φ.

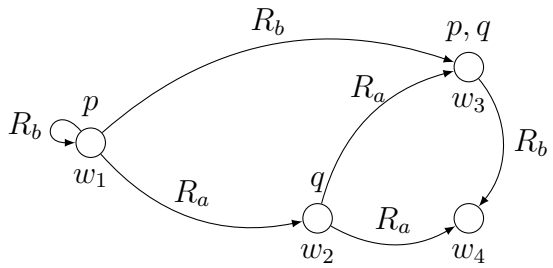
Of course this is not a minimal definition. One can fix an adequate set of primitive boolean connectors (like $\neg$ and $\wedge$) and define all the other boolean connectors in terms of that primitive set. Also, as it will follow from the satisfaction definition we are going to present below, diamond and box are dual operators, and therefore for all $r \in \textsc{rel}$, $\langle r \rangle \varphi$ can be defined as $\neg[r]\neg\varphi$, and conversely, $[r]\varphi$ is equivalent to $\neg\langle r \rangle \neg\varphi$. We are not going to bother yet to pick a set of primitives operators, since it is not really important at this point. When we do that, we will only have to worry about choosing a convenient set that allows us to generate the whole language.

Now we formally define the models for the basic modal language. As we mention before, Kripke semantics define models as graphs, and in fact, as *directed* graphs with *decorations*.

1.2.2. Definition. [Kripke models] Let $\mathcal{S} = \langle \textsc{prop}, \textsc{rel} \rangle$ be a signature. A Kripke model $\mathcal{M}$ for $\mathcal{S}$ is a tuple $\langle W, (R_r)_{r \in \textsc{rel}}, V \rangle$ satisfying the following conditions: (i) W, the *domain*, is a nonempty set whose elements are called points, but also, depending on the context, states, worlds, times, etc.; (ii) each R_r, an *accessibility relation*, is a binary relation on W and (iii) $V : \textsc{prop} \to 2^W$, the *valuation*, is a labeling function that assigns to each propositional symbol $p \in \textsc{prop}$ a subset of W. We can think of $V(p)$ as the set of points in $\mathcal{M}$ where p holds.

Given a model $\mathcal{M}$ and w in the domain of $\mathcal{M}$, we call $\langle \mathcal{M}, w \rangle$ a *pointed model*.

Before moving on, let's see an example of a Kripke model, in order to clarify the concept. Consider the following model $\mathcal{M} = \langle W, (R_r)_{r \in \textsc{rel}}, V \rangle$:



This model has a domain of four points, $W = \{w_1, w_2, w_3, w_4\}$. The signature in which it is based on is $\langle \textsc{prop} = \{p, q\}, \textsc{rel} = \{a, b\}\rangle$, that is, it has two binary relations, R_a and R_b, and two propositional symbols, p and q. We explicitly indicate in the picture the places where the propositional symbols hold. Translated to the valuation function V, that means that $V(p) = \{w_1, w_3\}$ and $V(q) = \{w_2, w_3\}$. Observe that at w_4 no propositional symbol holds.

If we look at the definition of Kripke models again, we can see that the directed graph itself is defined by the domain and the accessibility relations, whether the valuation function plays the role of *decorating* the graph with propositional symbols. Given a model $\mathcal{M}$, we call the *frame* underlying $\mathcal{M}$ its first two components, that is, $\langle W, (R_r)_{r\in\textsc{rel}}\rangle$. When dealing with models with only one accessibility relation, we often omit the set $(R_r)_{r\in\textsc{rel}}$ and we write directly $\langle W, R, V\rangle$ for the model and $\langle W, R\rangle$ for the frame.

Sometimes we are going to be interested in restricting ourselves to a portion of a given model. Let's define here the notion of *submodel*.

1.2.3. Definition. Let $\mathcal{M} = \langle W, (R_r)_{r\in\textsc{rel}}, V\rangle$ and $\mathcal{M}' = \langle W, (R'_r)_{r\in\textsc{rel}}, V'\rangle$ be two models. We say that $\mathcal{M}'$ is a *submodel* of $\mathcal{M}$ if $W' \subseteq W$, each R'_r is the restriction of R_r to W' (that is, $R'_r = R_r \cap (W' \times W')$) and V' is the restriction of V to $\mathcal{M}'$ (that is, for each $p \in \textsc{prop}$, $V'(p) = V(p) \cap W'$).

Now we are ready to define the semantics for the basic modal language, since we already have both the syntax and the structures the language is going to talk about. Recall that modal logics describe Kripke structures *from an internal perspective*. This means that, in contrast with first order logic in which formulas see models from some kind of omniscient lookout point, modal formulas are evaluated *at some particular point* of the model.

1.2.4. Definition. Given the model $\mathcal{M} = \langle W, (R_r)_{r\in\textsc{rel}}, V\rangle$ and $w \in W$, we inductively define the notion of a formula φ being satisfied (or *true*) in $\mathcal{M}$ at the

point w as follows:

$$
\begin{array}{rcl}
\mathcal{M}, w \models \top & & \text{always} \\
\mathcal{M}, w \models \bot & & \text{never} \\
\mathcal{M}, w \models p & \text{iff} & w \in V(p) \quad p \in \textsc{prop} \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \varphi \vee \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ or } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \varphi \rightarrow \psi & \text{iff} & \mathcal{M}, w \not\models \varphi \text{ or } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \varphi \leftrightarrow \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ if and only if } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \langle r \rangle\varphi & \text{iff} & \text{there is a } w' \text{ such that } wR_r w' \text{ and } \mathcal{M}, w' \models \varphi \\
\mathcal{M}, w \models [r]\varphi & \text{iff} & \text{for all } w' \text{ such that } wR_r w', \mathcal{M}, w' \models \varphi
\end{array}
$$

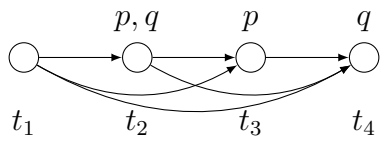
Given a model $\mathcal{M}$, we say that φ is *globally satisfied* (or *globally true*) on $\mathcal{M}$, and write $\mathcal{M} \models \varphi$, if for all points w in the domain of $\mathcal{M}$ we have that $\mathcal{M}, w \models \varphi$. A formula φ is *valid* if it is globally satisfied in all models, and in that case we write $\models \varphi$. A formula φ is *satisfied in a model* $\mathcal{M}$ when there is a point in $\mathcal{M}$ where φ is true, and φ is *satisfiable* if there is some point in some model at which it is satisfied. When working with sets of formulas, these definitions are lifted in the expected way. Given a class of models $\mathcal{C}$, we will note $T(\mathcal{C})$ the set of all valid formulas in $\mathcal{C}$. Given a model $\mathcal{M}$ and a point w in $\mathcal{M}$, we denote $T(\mathcal{M}, w) = \{\varphi \mid \mathcal{M}, w \models \varphi\}$. A formula φ is a *local semantic consequence* of a set of formulas Σ if for all models $\mathcal{M}$ and all points $w \in \mathcal{M}$, if $\mathcal{M}, w \models \Sigma$, then $\mathcal{M}, w \models \varphi$. When that is the case we write $\Sigma \models \varphi$. We define the set of consequences of a set of formula as $\mathsf{cons}(\Sigma) = \{\psi \mid \Sigma \models \psi\}$. Sometimes we will use this definition applied to a single formula instead of a set, with the usual interpretation. In Chapter 2 we are going to define the notion of *global semantic consequence*, but for now let's stay with the local version of the definition. When we say just "semantic consequence" we always refer to local semantic consequence.

One final remark about the basic modal logic. In the same way we define a language that supports multiple modalities, there is no reason to restrict ourselves to unary modalities. An extended version of the basic modal language can be given, in which modalities are n-ary, and each one is interpreted by an $n+1$-ary relation on the domain. We choose to define it in a more restricted manner in order to keep the notation compact, and also because we are not really going to exploit that specific feature of modal languages.

1.2.1 Some modal examples

We have already defined a number of concepts that allow us to start digging in the modal logic universe. Let's consider now some examples in order to grasp a general flavor of how modal languages work.

1.2.5. Example. Consider the following model:

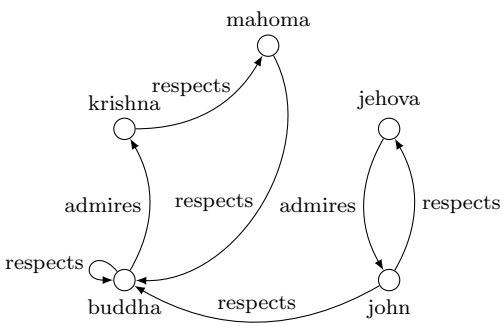


This model represents four different instant in time, and the accessibility relation is the "earlier than" relationship, that orders the flow of time (and consequently, it is a transitive relation). If we want to evaluate a formula in this model we first have to decide in which instant of time we want to evaluate it, and at each point the result of the evaluation may varies. For example, the formula $\Diamond p \wedge \Diamond q$ is satisfied at t_1 and t_2, because both points are before t_3 and t_4, where p and q hold respectively. But $\Diamond p \wedge \Diamond q$ does not hold at t_3, and neither at t_4. On the other hand, consider the formula $\Diamond(p \wedge q)$. In contrast with the previous formula, note that this one only holds at t_1, since it is requiring the existence of a future point in time where p and q hold *simultaneously*. One particular situation is the case of t_4. In this point, the formula $\Box p$ is true. Why? Because if we check the satisfaction definition for the $\Box$ operator, we can see that $\Box\varphi$ holds at a point w iff φ holds at *all* the successors of w. Since t_4 has no successors, this condition is trivially true. In fact, this means that $\Box\varphi$ will be true at t_4 no matter which φ we choose (and even when φ is $\bot$).

The idea of using modal logics to model events in time is due to Prior (see [Pri67, Pri77]), and he mainly analyzed the use of tenses relative to the point of speech. Nowadays, the study of temporal logics is very well developed [All05, Eme90, Bla90, vB91, HR06], and briefly speaking, modal temporal languages are usually interpreted on models that have a "time-like" structure. This can be a linear structure (like the toy example we have just presented), or some kind of transitive tree, representing branching time.

The next example shows that modal logics are not only a way to model "intended meanings", but also that they can be used as a tool to represent knowledge and to describe extensional information.

1.2.6. Example. Let's analyze this group of people and their relations:



Note that in this model we have two different modalities: *admires*, and *respects*. There are many things we can say here with modal formulas. For example, the formula

$$[\textsc{respects}]\bot \wedge \langle\textsc{admires}\rangle\langle\textsc{respect}\rangle\top$$

is only true when evaluated at Jehova: he does not respect anyone, but he admires John, who respects Buddha.

Nowadays modal logic is widely used to describe extensional situations. In fact, there is a "flavor" of modal logics, called *description logics* (DLs) that are specially designed to represent concepts, individual elements and their relations. Description logics were born in the computer science community, in contrast with classical modal logics, who came from the philosophy/mathematical area. Furthermore, there are available many inference tools for DLs that make them very useful tools in practical applications. More references about descriptions logics can be found in [BCPS07].

Proposition dynamic logic is a very nice example to present at this point. On one hand, it involves reasoning about execution traces of programs, and allows to explicitly model its inductive structure. On the other hand, it is an example of a modal language that is not a strict fragment of first order logic, and nevertheless, it is still decidable!

1.2.7. Example. The language of propositional dynamic logic (PDL) has an infinite collection of modalities. Each modality $\langle\pi\rangle$ represents a non-deterministic program, and the intended interpretation of $\langle\pi\rangle\varphi$ is 'some terminating execution of the program π from the current state leads to a state carrying the information φ'.

What is interesting about PDL is that the structure of programs is made explicit in the syntax. When writing a formula, complex programs can be built using primitive ones with the help of a set of programs constructors. Let's suppose fixed a set of basic programs π_a, π_b, π_c, and so on (that is, we have the modalities $\langle\pi_a\rangle, \langle\pi_b\rangle, \langle\pi_c\rangle, \ldots$ as part of the language). With these basic programs we can construct new programs in the following way:

- (**composition**) If π_1 and π_2 are programs, then $\pi_1;\pi_2$ is also a program. The program $\pi_1;\pi_2$ first executes π_1 and then π_2.
- (**union**) If π_1 and π_2 are programs, then $\pi_1 \cup \pi_2$ is also a program. The program $\pi_1 \cup \pi_2$ executes non-deterministically π_1 or π_2.
- (**iteration**) If π is a program, then $\pi^\star$ is also a program. The program $\pi^\star$ executes π a finite (zero or more) number of times.

Given the connection we have between programs and modalities, this means that if $\langle\pi_1\rangle$ and $\langle\pi_2\rangle$ are modalities, then $\langle\pi_1;\pi_2\rangle$, $\langle\pi_1 \cup \pi_2\rangle$ and $\langle\pi_1^\star\rangle$ are also

modalities. Observe that in PDL there are two layers of syntax, one inner layer for modalities, that allow us to describe the structure of programs, and an external layer that uses those modalities to characterize the behavior of program executions. For example, the formula $p \to \langle(\pi_1 \cup \pi_2)^\star\rangle q$ says that if we are in a state where p holds, then a state where q holds can be reached executing π_1 or π_2 a finite number of times.

The three ways of constructing new programs (composition, union and iteration) constitute what it is called *regular* PDL. However, there are others constructors that can be added, for example:

- (**test**) If φ is a formula, then $\varphi?$ is a program. The program $\varphi?$ tests whether φ holds, and if so, continues. If φ does not hold, it fails.

Observe that the test operator allow us to turn a formula into a modality. This constructor, combined with other programs, can produce very interesting examples. For example the formula

$$\langle(\varphi?; a)^\star; (\neg\varphi)?\rangle\psi$$

represents that the program "while φ do a" ends in a state satisfying ψ.

The idea to extend modal logics with modalities representing programs is due to Pratt [Pra76], and PDL itself was first investigated by Fischer and Ladner [FL77, FL79]. The capability to perform finite iterations, through the construction $\pi^\star$, provides PDL with a way to express the transitive closure operator. This means that PDL has some kind of second-order expressivity, beyond first order, since first order logic cannot define the transitive closure. Surprisingly, PDL is decidable, and its satisfiability problem is EXPTIME-complete [FL77, FL79, Pra79].

The last example we are going to discuss shows how modal logics can be applied to model knowledge and belief, that is, we want to show that modal logics can also be used in the field of *epistemic logics*.

1.2.8. Example. Epistemic logics deal with agents and what agents consider possible given their current information. Suppose we have three card players: John (j), Peter (p) and Carol (c), and each player holds one of the cards with colors *green* (g), *blue* (b) and *red* (r). In the game they are playing, each of the three have looked at his own card, but have kept it hidden from the others. This situation can be modeled as follows: the proposition p_c represents a situation where the player p holds the card c. For example, $John_{blue}$ (abbreviated j_b) indicates that John holds the *blue* card. The inability of a player p to distinguish two different situations of the game, given his current knowledge, is represented with a modality $\langle p\rangle$. That is, if the player p cannot distinguish between the point

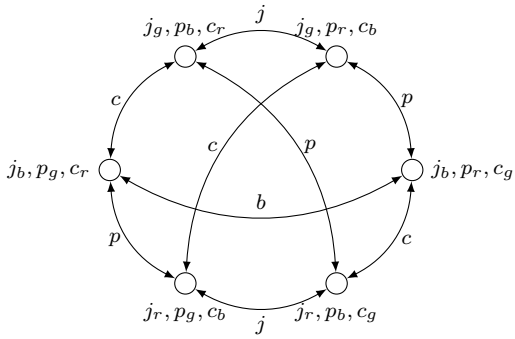


Figure 1.1: The model $\mathcal{M}$, representing possible deals

w and w', then w is connected with w' through the relation p. If we represent every possible deal in the game, the result is the model shown in Figure 1.1.

For example, imagine that the current deal is: John holds the green card, Peter the blue card and Carol the red card. Looking at the model, we are in the point j_g, p_b, c_r. Let's analyze what John can say in this situation. He knows he holds the green card, but he doesn't know which cards the other players have. One possibility would be the real one (in which Peter has the blue card and Carol has the red card), but the other possibility for John is that Peter has the red card and Carol the blue card. He cannot distinguish between these two situations, and we can express it with the formula $j_g \wedge p_b \wedge c_r \rightarrow \langle j \rangle (j_g \wedge p_r \wedge c_b)$. Observe that this formula is globally satisfied in our model.

Epistemic logics can also be dynamic, in order to reflect the changes in the information the agents have. *Public announcements* is a way of communication between agents, where an agent makes some information public, so everybody knows it. So, if an agent announces φ, all the points where $\neg\varphi$ holds should be *erased* from the model, since they are no longer compatible with the information that all the agents have. Let's suppose that John decides to show his card to the other two players. After that action, everybody knows that John holds the green card, and therefore all the points in the model where John holds a card different than green are no longer possible. That action transforms the previous model into this one:

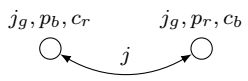


Figure 1.2: The modified model $\mathcal{M}'$, after John shows the green card

That is, in the resulting model $\mathcal{M}'$, Peter and Carol knows all the cards and therefore they can distinguish between the two situations left. Therefore, there

are neither p-relations nor c-relations in $\mathcal{M}'$. On the other hand, John only knows that he has the green card, so he still cannot say which are the cards of the other players. That is why there is a j-relation connecting the two situations (in both directions).

Epistemic logics were first defined by Hintikka [Hin62], but there is also more recent work available, such as [FHMV03] by Fagin, Halpern, Moses and Vardi. There is also much work in dynamic epistemic logics, and the notion of model update, see [BB99, Ger99, vBL07, vB06, vBvEK06, Pla07].

1.3 A wide menu to choose from

So far we have been dealing mostly with the basic modal language, and we showed some examples that illustrated a general horizon of possible uses. At this point we might be tempted to ask: is it possible to enrich modal logics to obtain stronger languages keeping at least some of its attractive properties? Since we were talking about the *basic* modal language, we can suspect that the answer to this question is positive. Modal logicians have been investigating richer languages for many years, so here we are going to show some of their work.

The process of enriching the basic modal language can take many directions. Sometimes it consists of insisting that a modality should be interpreted by some specific relation (the universal modality is a good example, as we will see). Occasionally the enrichment is designed to support new semantics capabilities, as we are going to show with hybrid logics or propositional dynamic logic. In other cases, it takes the form of more complex satisfaction definitions, like temporal logics and Since and Until operators. There are also many other ways in which the modal languages can be extended. The existence of this motley extensions naturally rises the question: what do all these richer languages have in common? What are the properties that make them 'modal'? This is not an easy question, but there are some recurrent notions in all these languages: restricted quantification, characterizations of fragments of first and second order logic, and decidability in most of the cases.

1.3.1 The universal modality

We have emphasized the locality of modal logics, but sometimes there are situations that demand a global perspective. For example, suppose we are working with a modal language for talking about the weather, and in this language φ means "we are in summer", and ψ means "the temperature is beyond 25 degrees". With the basic modal language there is no clear way to express that *whenever* we are in summer, the temperature is beyond 25 degrees. If we check for the satisfiability of $\varphi \to \psi$ and we get a positive answer, this only means that there is a model and a point where φ is false or ψ is true. But we want to force

every point in the model to satisfy $\varphi \to \psi$. This is a case where the universal modality is useful.

Let's suppose that we are working with only one modality, just for the sake of simplicity. We can add to this language a second modality called E whose interpretation is fixed: in every model $\langle W, R, V\rangle$ the interpretation of E must be the universal relation $W \times W$. The formal semantic definition is the following:

$$\mathcal{M}, w \models E\varphi \quad \text{iff} \quad \text{there is a } u \in W \text{ such that } \mathcal{M}, u \models \varphi.$$

So this new modality scans the whole model looking for a point that satisfy φ. Its dual $A\varphi \equiv \neg E \neg \varphi$ has the following interpretation:

$$\mathcal{M}, w \models A\varphi \quad \text{iff} \quad \text{for all } u \in W \text{ we have that } \mathcal{M}, u \models \varphi.$$

That is, $A\varphi$ checks if φ holds at all points in the model. Observe that in this way we have *internalized* the notion of global truth into the modal language: for any model $\mathcal{M}$ and any formula φ, we have that $\mathcal{M} \models \varphi$ iff $A\varphi$ is satisfiable in $\mathcal{M}$.

So now we can formulate our question about the weather using the modal language extended with the universal modality. To test whether whenever we are in summer, the temperature is beyond 25 degrees, we can check the satisfiability of the formula $A(\varphi \to \psi)$.

Universal modality seems to be a nice operator to have, but what happened with what we said about modal languages providing an "internal perspective" of the structure of models? Have we broken this principle including the universal modality? The first impression could point in that direction, but in fact the extended language still takes an internal perspective. The universal modality places modal formulas in every possible point in the model, but formulas are *still* evaluated at a particular point. For example, this is reflected in the fact that we still preserves decidability. In [Hem96], Hemaspaandra showed that the basic modal language enriched with the universal modality is EXPTIME-complete.

1.3.2 Since and Until operators

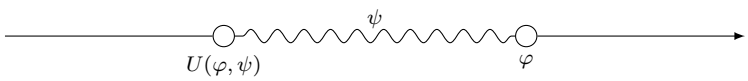
Recall Example 1.2.5, in which we mention temporal logic, and the work of Prior with tense logic. In the late 1960s the operators U (Until) and S (Since) were added by Hans Kamp [Kam68] to Prior's logic. Later, around 1980, Gabbay, Pnueli, Shelah and Stavi [GPSS80] realized that these operators were extremely useful to reason about program executions, since they allow to describe the conditions that an execution must satisfy all along its life-cycle.

Even though these operators are mostly used in temporal logic, we define them here in their most general form:

$$\begin{array}{lll} \mathcal{M}, w \models U(\varphi, \psi) & \text{iff} & \text{there is a } v \text{ such that } wRv \text{ and } \mathcal{M}, v \models \varphi, \text{ and} \\ & & \text{for all } u \text{ such that } wRu \text{ and } uRv, \mathcal{M}, u \models \psi. \\ \mathcal{M}, w \models S(\varphi, \psi) & \text{iff} & \text{there is a } v \text{ such that } vRw \text{ and } \mathcal{M}, v \models \varphi, \text{ and} \\ & & \text{for all } u \text{ such that } vRu \text{ and } uRw, \mathcal{M}, u \models \psi. \end{array}$$

Intuitively, if we think about the meaning of these operators in the context of temporal logics, *Until* checks if there is some point v in the future where φ holds, and that at all points between the point of evaluation and v, ψ holds. *Since* works similarly but looking towards the past.

The first thing to ask is what can we say with these operators. To start with, we should notice that they can say at least the same things as regular diamonds. Why? Observe that $U(\varphi, \top)$ is semantically equivalent to $\Diamond\varphi$. But with S and U we can say more. In the analysis of program executions, it is often needed to express properties like "something good is going to happen, and until that time nothing bad will happen". Such properties are often called *guarantee properties* in the computer science community. *Since* and *Until* seem perfectly tailored to express these kind of things. Let's take a temporal linear model, with the usual relation "earlier than", that orders the flow of time. If φ represent a good event, and ψ the guarantee that nothing bad is happening, then $U(\varphi, \psi)$ is expressing a guarantee property in terms of φ and ψ.



As we said before, modal logics are robustly decidable, and the extension with *Since* and *Until* is not an exception. The basic modal logic extended with *Since* and *Until* is decidable, and the most general definition falls into what is called the *packed fragment* [Mar01], a syntactic restriction of first order logic, whose satisfiability problem is 2EXPTIME-complete. When the class of models is restricted to linear orders, then there are better complexity bounds, see [Rey03, DR07]

1.3.3 Hybrid Logics

As we said before, one of the cornerstones of modal logics is the *internal perspective* they provide to inspect models. The curious thing about this matter is that even though points in a model play a fundamental part in modal logics semantics, the basic modal languages cannot directly name them. The feeling is that, when evaluating a formula in a model, the points are *hidden* from the syntax. We cannot say that some *particular* individual has some property, or that two different processes lead us to the *same* state. Of course this is not the case for first order logic, where we can use constants to *name* individuals of interest, and the

equality symbol to check their identity. The *basic hybrid language* is the result of addressing this issue. They treat points in the model as first class citizens, using an special sort of atomic formulas to refer to points. The first steps in this direction were taken by Prior [Pri67, Pri77] in the 1960s, when he first proposed to sort propositional symbols and use formulas as terms.

So let's take a modal language with propositional symbols p, q, r, and so on, and add a second sort of propositional symbols NOM, called *nominals*, typically written i, j, k, etc. Nominals can be used to construct new formulas in the same way propositional symbols do, but the key difference is that we are going to add the constraint that *each nominal must be true at exactly one point in any model.* In this way, nominals *name* points. From the formal point of view, for any model $\langle W, (R_r)_{r\in\text{REL}}, V\rangle$, the valuation function V is extended to interpret also nominals, $V : \text{PROP}\cup\text{NOM} \to 2^W$, and we insist that $V(i)$ is a singleton set, for any valuation V and nominal i. Observe that we are not asking that at every point in the model a nominal must hold. It is perfectly right if we have points without name.

This change already increases the expressive power. For example, the formula

$$\Diamond(i \wedge p) \wedge \Diamond(i \wedge q) \to \Diamond(p \wedge q)$$

is valid: if the antecedent is satisfied, that means that at the (unique) point where i holds, the propositions p and q also hold. This is clearly not the case if we replace i by some arbitrary propositional symbol.

Once we have added names to the language, *satisfaction operators* turn up quite naturally: we would like to have a way to evaluate a formula at a specific named point. Satisfaction operators has the form $@_i\varphi$ (read 'at i, φ') where i is a nominal. The formula $@_i\varphi$ moves the point of evaluation to the point named by i, and evaluate φ there. More formally, given a model $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V\rangle$, the semantics is the following:

$$\mathcal{M}, w \models @_i\varphi \quad \text{iff} \quad \mathcal{M}, u \models \varphi, \text{ where } V(i) = \{u\}.$$

Satisfaction operators can be thought of as modalities, but observe that they are self-dual: $@_i\varphi \leftrightarrow \neg @_i \neg\varphi$. That is, we can think of satisfaction operators as either diamonds or boxes.

There is an issue that can be confusing the first time one starts using these operators. Which is the difference between the formulas $i \wedge \varphi$ and $@_i\varphi$? At first glance, both seem to say that φ holds in the point named by the nominal i. The key distinction is that $@_i\varphi$ holds *independently* of the point in which it is evaluated, since the satisfaction operator actually *moves* the point of evaluation to the point named by i. This is clearly not the case for $i \wedge \varphi$, that only checks whether the point of evaluation is the one named by i, and if φ holds there.

There is another perspective that justifies the existence of satisfaction operators. Recall what we said about the use of constants in first order logic to name individuals, combined with the use of the equality symbol to compare them. In

the same way, satisfaction operators give us a *modal equality* to compare individuals. Note that formulas like $@_i j$ asserts that the point named by i is identical to the point named by j. In this way, we can establish a modal theory of equality: for all nominals i, j, the formulas $@_i i$ (reflexivity), $@_i j \to @_j i$ (symmetry), $@_i j \wedge @_j k \to @_i k$ (transitivity) and $@_i \varphi \wedge @ i_j \to @_j \varphi$ (congruence) are valid.

The idea of *sorting* atomic formulas, and to use one sort of atoms to refer to points in the model is quite simple. This simplicity gets reflected in hybrid logic complexity: we don't have to pay a special price for this additional machinery, and the satisfiability problem for the basic hybrid logic remains PSPACE-complete [ABM01]. And even more, in certain aspects hybrid logic behaves better than its modal counterpart. Completeness is a good example of this, and nominals can be very helpful in many situations to provide a nice completeness theory.

The next natural step is to think of nominals not as names, but as variables over individual points, and to add quantifiers. It is quite clear that this path gets us even closer to classical logics, in which quantifiers and variables are explicit in the syntax. But this is one of the nice features of modal logics: they allow to increasingly add new features to a logic, but in a fine-grained way, taking care of what are the aspects that are being incorporated. The classical first order notion of quantifiers does not reflect the intrinsically local behavior of modal logic. That is the motivation to introduce $\downarrow$, the downarrow binder. This binder allow us to create a name "on the fly" for the current point of evaluation, and let us refer to it later in the formula. That is, when evaluating $\downarrow x.\varphi$ in a point w, the variable x will act in φ as a nominal that names w. For example, in any model $\mathcal{M}$. the formula $\downarrow x.\Diamond x$ is true precisely at the reflexive points.

Since now we have variables, to formally define the semantics of $\downarrow$ we need an assignment function g to evaluate a formula. This is equivalent to the assignment function for first order logic. We don't necessarily need a new sort of atomic symbols to represent variables, we can reuse nominals and think that the binder 'resets' a nominal to the current point of evaluation. In this scenario, the valuation function V does not interpret nominals anymore, since this task is now delegated to g. So, given a Kripke model $\mathcal{M} = \langle W, (R_r)_{r \in \text{REL}}, V \rangle$ and an assignment function $g : \text{NOM} \to W$, the semantics conditions for downarrow are defined as follow:

$$\begin{array}{rcl} (\mathcal{M}, g), w \models i & \text{iff} & g(i) = w \\ (\mathcal{M}, g), w \models \downarrow i.\varphi & \text{iff} & (\mathcal{M}, g^i_w), w \models \varphi \text{ where } g^i_w \text{ is identical to } g \\ & & \text{except perhaps in that } g^i_w(i) = w. \end{array}$$

There is a nice interplay between $\downarrow$ and @. We can think of $\downarrow$ as storing values for nominals in the assignment function, while @ retrieves them. Consider the following formula, which is true in any model at points where the accessibility relation behaves transitively.:

$$\downarrow i.\Box\Box\downarrow j.@_i\Box j.$$

This formula first names the point of evaluation i, and then names j all the two-steps successors of i. Then, using @, it jumps back to i to assert that all the one-step successors of i are also two-step successors.

As we can see, $\downarrow$ provides a significant increment of expressivity. This additional expressive power comes with a price: we have crossed the border of decidability. Already the satisfiability problem for the hybrid language with nominals and $\downarrow$ (that is, even without @) is undecidable [BS95] (and originally in unpublished work by V. Goranko). But this logic is not as expressive as first order logic. For further details about the characterization of this fragment see [ABM01].

1.4 Modifying models: memory and state

In this section we will start heading to the main subject of this thesis. We are going to present a new family of modal logics which we call *memory logics*. Let's recall the analogy of thinking about a modal formula as an automaton standing at some point, and only permitted to locally explore the structure of the model. But what about "changing" the model? Suppose we want to grant our little automaton the additional power to *modify* the structure during its exploratory trips. This question is not new, and it has resulted in different proposals of what are called *dynamic* logics.

Consider, for example, the task of assigning semantics to a programming language. Clearly, the different instructions of the language change the computational state. It is then natural to specify their semantics by defining exactly which changes each atomic operation of the language introduces. This idea is at the core of logics like Hoare-Floyd logics [Hoa69, Flo67] which include, for example, special operators to indicate the state of variables before and after a given instruction.

As a second example, consider the area of linguistics called dynamic semantics. One of its fundamental claims is that the standard truth-conditional view of sentence meaning – which is the result of using classical logic as representation languages – does not do sufficient justice to the fact that uttering a sentence changes the context it was uttered in. Deriving inspiration, in part, from work on the semantics of programming languages, dynamic semantic theories have developed several variations on the idea that the meaning of a sentence is to be equated with the changes it makes to a context. Different dynamic logics like the ones introduced in, e.g., [GS91a, GS91b] try to capture these ideas.

As yet a third example with an ample literature we could cite different dynamic epistemic logics [vDvdHK07, vBvEK06, vB01, Pla07, Ger99, vB05]. As we saw in the Example 1.2.8, these are languages used to model the evolution of the knowledge of epistemic agents via *updates* to the model representing their epistemic state. The action of a public announcements we saw there can be carried out by an operator that updates the model, eliminating all alternative epistemic

states that are incompatible with the new provided information.

Our last family of examples come from the area of temporal logics for verification. In this area, it is many times necessary to model time-critical systems that depend on quantitative rather than qualitative properties. Many temporal logics introduced for this task use explicit global clocks which are accessed and controlled through logical operators. Examples of such logics are XCTL [HLP90], half-order logics [AH89a, Hen90], and timed and metric temporal logics [ACD93, AFH96, Koy90, OW05].

On the other hand, other dynamic logics are not dynamic in the sense mentioned above. In PDL, for example, formulas are evaluated in a model but they cannot modify it.

Memory logics can be seen as an attempt to investigate some of the common characteristics of all these logics, in the simplest possible set up. Going back to our little automaton, suppose we extend our definition of a model to a 4-uple $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$, where S is an arbitrary subset of W. We can think of S as a memory where the automaton can store a point that it considers particularly interesting. Defining the semantics of this operation is straightforward. Suppose that we use $\textcircled{r}$ ('remember') to represent the memorize operator, then we could define

$$\langle W, (R_r)_{r\in\text{REL}}, V, S\rangle, w \models \textcircled{r}\varphi \text{ iff } \langle W, (R_r)_{r\in\text{REL}}, V, S \cup \{w\}\rangle, w \models \varphi.$$

Notice that $\textcircled{r}$ modifies the model, and that φ is evaluated in the modified model. The operation $\textcircled{r}$ by itself is totally useless. If we cannot access the information stored in S, $\textcircled{r}\varphi$ is equivalent to φ. Let's add then an operator $\textcircled{k}$ ('known') that checks whether the current point has been previously memorized:

$$\langle W, (R_r)_{r\in\text{REL}}, V, S\rangle, w \models \textcircled{k} \text{ iff } w \in S.$$

Even with this simple addition we have that $\langle W, (R_r)_{r\in\text{REL}}, V, \emptyset\rangle, w \models \textcircled{r}\langle r\rangle\textcircled{k}$ will be true only if w is self reachable via the accessibility relation corresponding to the modality $\langle r\rangle$. This property cannot be expressed in the basic modal language.

In the same spirit of the operators $\textcircled{r}$ and $\textcircled{k}$ introduced above, we can naturally define operators that modify any component of a model (adding or deleting nodes and edges or modifying the labeling function). In Chapter 2 we are going to explore these ideas with detail.

To sum up, memory logics are a novel family of modal logics that allows to model dynamic behavior through explicit memory operators that *change* the evaluating structure. This proposal introduces a framework for studying the notion of state in a more general way, without bounding the analysis to any fixed domain (like knowledge change, time flow, linguistics contexts, etc.). Most of the work that has been done in this direction implicitly adds some specific native behavior in the "dynamic component". We want to study some of the dynamic capabilities of the above mentioned approaches from a more abstract point of view, and analyze the different aspects of this family in terms of logic properties.

1.5 Overview of the thesis

In the previous section we gave an informal presentation of the family of memory logics. In the rest of this thesis we are going to see that the idea of adding an explicit memory to Kripke models, and operators that are able to query and modify that memory, extend the modal logic menu with new fragments with very interesting properties. This is a new family of logics, and therefore there are several aspects that are worth investigating.

The first thing we want to know about these logics is what can we actually *say* with them. Are they really providing additional expressive power, or they are just a complicated way of writing the same things we already could say with the basic modal language? Or perhaps it is the other way around, and adding an explicit memory places these logics beyond the scope of modal or even hybrid logics. To move one step at a time, in Chapter 2 we are going to formally introduce the family of memory logics, we will show some examples, and present the different fragments we are going to work with. Then in Chapter 3 we are going to study this family in terms of expressive power. In this chapter we want to show which is the impact of the different memory operators we consider, and how is the interaction between them. To be able to do that, we will introduce a key concept to analyze expressivity: *bisimulations*.

After studying the expressive power, the next natural step is to look at the computational complexity. In Chapter 4 we will analyze some fragments of the memory logic family and we will determine whether its satisfiability problem is decidable for each case. Have we added enough expressive power to cross the border of decidability? To check that, we are going to introduce a technique called *tiling*, which is very useful to prove undecidability results. Also, we are going to show that some fragments have the *finite model property*, which is a key property to have decidability.

Then, in Chapter 5, we are going to analyze Craig interpolation and Beth definability for some memory logic fragments. A logic enjoying these properties can be seen as having "completeness in the theory of definitions", as opposed to the theory of deductions. These properties also allow reasoning systems to be set up in a modular way.

We are also interested in studying memory logics from a proof theoretical perspective. In Chapter 6 we are going to turn to Hilbert style axiomatic systems, and we are going to try to characterize several fragments of the memory logic family mostly using techniques borrowed from hybrid logics. In this chapter we are going to see that nominals enable us to provide simpler axiomatic systems in many situations.

Finally, in Chapter 7 we are going to study semantic tableau systems for memory logics. Taking inspiration from the prefixed tableau calculus developed for hybrid logics, we are going to present a sound and complete tableaux calculi that works with a rich memory logic fragment, and some of its sub languages.

We close in Chapter 8 with some concluding remarks, open problems and further work.

Many of the results presented along this thesis were published in [AFM09, AFFM08, AFGM09].

Chapter 2

Memory Logics

Sabía las formas de las nubes australes del amanecer del treinta de abril de mil ochocientos ochenta y dos y podía compararlas en el recuerdo con las vetas de un libro en pasta española que sólo había mirado una vez y con las líneas de la espuma que un remo levantó en el Río Negro la víspera de la acción del Quebracho.

"Funes el memorioso", Jorge Luis Borges.

In the previous chapter we gave some intuitions about memory logics, and the idea of explicitly adding state to Kripke models. As we said before, this idea is not new, but most of the work that has been done in this direction is for domain-specific reasons. We want to analyze the idea of incorporating the notion of state from a "pure" perspective, in which we don't add any specific built-in behavior in the evolution of the model (like clocks, programming language semantics, epistemic updates, etc. do). As we mention in Section 1.4, we are going to do this by analyzing which is the outcome of adding a memory to a Kripke model. Having a memory and specific operators to access and change it, provides us with a nice framework for studying dynamic logics from this perspective.

2.1 Focusing on sets

What kind of memory do we want to have? Of course this choice will have a major impact in the type of logic that will come out. Also the type of memory to use will determine the set of possible operators that will interact with it, in order to retrieve or write data. We want to start analyzing this concept from their roots, and to accomplish that we will take the idea we presented in Section 1.4, and concentrate mostly in a very simple memory structure: a set that stores points of the model.

Let's see an example of this. Suppose that we have a Kripke model $\mathcal{M} = \langle W, R, V, S\rangle$ equipped with a memory as in Figure 2.1. Assume that $V(p) = \emptyset$ for all $p \in \textsc{prop}$ and that we have a unique relation R, as shown in the picture. The initial memory of this model in empty ($S = \emptyset$), and when a point is added to the memory we are going to graphically represent that with a solid black point (in contrast with the outlined white points we have now in the picture).

There are many things we can say here with the modal language extended with the operators ⓡ and ⓚ we presented in Section 1.4. Let's suppose fixed the

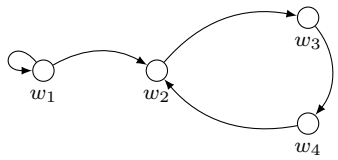
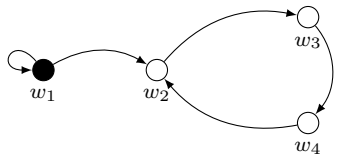


Figure 2.1: A model with an initially empty memory

model in Figure 2.1, and let's see how we can describe it using this language. For example, how can we say that w_1 has a successor different from itself? We can check whether the formula

$$\textcircled{r}\Diamond\neg\textcircled{k}$$

is satisfiable in w_1. What is this formula doing? First, it uses $\textcircled{r}$ to remember the current point of evaluation, adding w_1 to the memory. That means that the previous model turns into $\mathcal{M}' = \langle W, R, V, S = \{w_1\}\rangle$:



So now the remaining formula $\Diamond\neg\textcircled{k}$ is evaluated in $\mathcal{M}'$. To be satisfiable, this formula needs a w_1-successor v such that $v \notin S$. But this is the case, since w_2 fulfills that requirement. Observe that if we had not memorized w_1, then we would not be sure if the w_1-successor is in fact different from w_1.

We can also check, for example, whether w_2 is involved in a cycle of length 3. And even more, we can check that from the perspective of w_1. How can we do that? In the first place, we want to be sure that we actually move from w_1 to w_2, in order to avoid looping in w_1. To do that, we can use the same trick that we used before: we should remember w_1 and ask for a non-memorized successor. Then we should identify the cycle. The way to recognize it involves remembering also w_2, so we know when we are back to the same point. Therefore, to check this property we can evaluate the formula

$$\textcircled{r}\Diamond(\neg\textcircled{k} \wedge \textcircled{r}\Diamond\Diamond\Diamond\textcircled{k})$$

at w_1. Observe that this formula works well to check the property we wanted in *this particular model*, but it is easy to imagine other models that satisfy the same formula where there is not a cycle of length 3. For example, consider the model shown in Figure 2.2. Observe that the formula $\textcircled{r}\Diamond(\neg\textcircled{k} \wedge \textcircled{r}\Diamond\Diamond\Diamond\textcircled{k})$ also holds when evaluated at w_1 in that model. Therefore, this formula is not *characterizing* the models with cycles of length 3. In contrast, if we return to the previous example where we had the property "w_1 has a successor different from itself", the

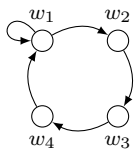


Figure 2.2: A model with a cycle of length 4

formula ⓡ$\Diamond\neg$ⓚ evaluated in a point w_1 distinguishes precisely the models that satisfy that property.

Notice that the fact that the memory was *initially empty* was crucial to express the properties we wanted. Intuitively, a non-empty memory adds "noise" when trying to express a property: every time we check if a point in the model is memorized, we are not really sure whether it was originally in the memory or it is the result of a previous application of the remember operator. That is why it is quite natural to consider starting to evaluate a formula with an empty memory, and it is over these models that the operators ⓡ and ⓚ have a clearer role. As we will see in the next chapters, the restriction to this class of models has important effects on expressivity and decidability. In Section 2.1.3 we are going to define this class of models formally.

2.1.1 Extended operators

As we saw, ⓡ stores points in the memory, and ⓚ allows to test for membership. The next step is to start thinking in operators that *delete* elements from the memory. The ability to erase points can be useful in many situations, as we will see. In the previous example, the initial memory of the model was completely empty, and this was quite convenient for our purposes, but imagine that the initial model is like in Figure 2.3. That is, all the points in the model are memorized.

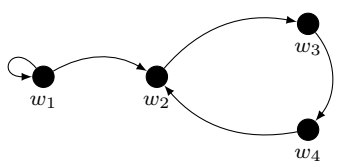


Figure 2.3: A model in which all the points are memorized

How do we say now that w_1 has a successor different from itself? The formula ⓡ$\Diamond\neg$ⓚ is no longer satisfiable in $(\mathcal{M}, w_1)$, since there is not a point where $\neg$ⓚ holds. Intuitively, it is quite clear that in this context there is no way of saying what we want, and in fact in Chapter 3 we are going to prove this formally. The situation changes if we add operators to delete elements from the memory. Let's

suppose that we add the operator ⓔ (for 'erase'), that completely wipes out the memory. Formally, its semantics is defined in the following way:

$$\langle W, (R_r)_{r\in\mathrm{REL}}, V, S\rangle, w \models ⓔ\varphi \quad \text{iff} \quad \langle W, (R_r)_{r\in\mathrm{REL}}, V, \emptyset\rangle, w \models \varphi.$$

Having ⓔ, it is easy to express that w_1 has a successor different from itself, even when the memory is not initially clean. We just have to prefix the formula we used in the first example with an ⓔ. That is, the formula

$$ⓔⓡ\Diamond\neg ⓚ$$

evaluated at w_1 expresses what we want: first, it cleans the memory using ⓔ, and then the remaining formula $ⓡ\Diamond\neg ⓚ$ is evaluated in the model we already saw in Figure 2.1.

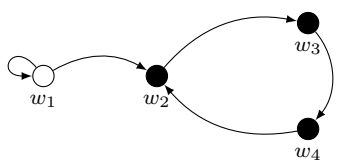
Observe that ⓡ has a local behavior, in the sense that it stores in the memory the *current* point of evaluation. In contrast, ⓔ has a global behavior, since it erases all the memory completely, independently of the evaluating point. This asymmetry suggests the idea of thinking in a local version of ⓔ, that deletes only the current point of evaluation. So let's also add the operator ⓕ (for 'forget'):

$$\langle W, (R_r)_{r\in\mathrm{REL}}, V, S\rangle, w \models ⓕ\varphi \quad \text{iff} \quad \langle W, (R_r)_{r\in\mathrm{REL}}, V, S \setminus \{w\}\rangle, w \models \varphi.$$

Taking again the model shown in Figure 2.3, in which all the points were memorized, we can use ⓕ to check, for example, whether w_1 is related to itself with the formula

$$ⓕ\Diamond\neg ⓚ.$$

This formula, evaluated at w_1, first deletes the current point from the memory, leaving the model like this:



And then the remaining formula $\Diamond\neg ⓚ$ is satisfied using the reflexive edge of w_1.

As we can see, even in this simple setup in which we take sets as the storage structures, there is a menu of different operators to choose. We are going to see that this menu increases if we add the capability of controlling the interaction between memory and modal operators.

2.1.2 Memorizing policies

Until now, memory and modal operators were working 'in parallel', in the sense that the logics we presented allowed to freely explore the model using modalities, and at the same time, to store or delete points from the memory during this exploration. As we said in Chapter 1, restricting expressivity can sometimes be helpful reducing computational cost, and that is why we can try to impose certain constraints in the interplay between memory and modal operators.

Let's see the following example to clarify these ideas. We mention in Section 1.2.1 that modal logics are useful to model knowledge and belief. Consider the model shown in Figure 2.4. A point in this model represents the epistemic state of an agent, that is, the agent's current knowledge. The neighborhood of a point w represents the possible "one-step" situations, modeling how the world might be from the perspective of w. This means that, for example, if the agent is in w_1, then w_0, w_1 and w_3 are epistemically acceptable alternatives to the current situation.

Suppose that we are interested in reasoning about the changes in the epistemic state of an agent a from a given point w, and we want to identify specifically which sequences of changes in the agent's knowledge leads again to w. This sequences can be thought of as an indicator that the knowledge of an agent is not really "evolving" when following certain paths. For example an agent may know p, but also accepts a possible situation where he does not know p anymore. This can be said with the formula $p \wedge \langle a \rangle \neg p$. He may also re-learn p later, and we can say this with $p \wedge \langle a \rangle (\neg p \wedge \langle a \rangle p)$. If a given model satisfies this formula, it means that the agent can start knowing p, and re-learn it again after some epistemic evolving steps. But how can we know if the rest of the agent's knowledge remains the same or not after all these changes?

For example, the model shown in Figure 2.4 allows both alternatives. The agent known p in w_0 and admits an alternative epistemic state w_1, where $\neg p$ holds. From w_1, the agent can learn p again returning to w_0, the original state, or he can travel to w_2. The epistemic state w_2 does not coincide with w_0: even though they agree on p, they differ on q.

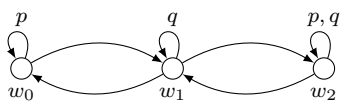


Figure 2.4: Epistemic states of an agent

We can use memory operators to distinguish these situations. The formula $p \wedge \textcircled{r}\langle a \rangle (\neg p \wedge \textcircled{r}\langle a \rangle (p \wedge \textcircled{k}))$ is true in models where p can be re-learned by returning to *exactly* the same epistemic state where the agent has started.

What is interesting from this example is that if we only care about identifying

"already visited" epistemic states, then every time we remember a point in the model is to perform a modal transition. That is, the actions of memorizing a point and exploring the graph are highly coupled. This suggests the idea that in this context perhaps we *don't need* $\langle r \rangle$ and ⓡ as two separated operators, but just one operator that performs both things at once. So let's introduce the $\langle\langle r \rangle\rangle$ operator that does exactly that:

$$\langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w \models \langle\langle r \rangle\rangle \varphi \quad \text{iff} \quad \begin{array}{l} \exists w' \in W, R_r(w, w') \text{ and} \\ \langle W, (R_r)_{r \in \text{REL}}, V, S \cup \{w\} \rangle, w' \models \varphi. \end{array}$$

In terms of expressivity, this operator is *a priori* weaker than having both ⓡ and $\langle r \rangle$, since now we cannot make a modal transition without memorizing the current point of evaluation. In the next chapters we are going to see that this memorizing policy in fact restricts the expressive power, and allows us to regain decidability for some fragments.

One can certainly imagine some other memorizing policies. For example, we can add operators in the line of dynamic epistemic logics: every time we make a modal transition, all the states that satisfy some semantic condition are added to the memory. The semantic condition may vary depending on the scenario. For example, we could define an operator $\langle\langle r \rangle\rangle_p$ like this: given a model $\mathcal{M} = \langle W, (R_r)_{r \in \text{REL}}, V, S \rangle$ and $w \in W$,

$$\mathcal{M}, w \models \langle\langle r \rangle\rangle_p \varphi \quad \text{iff} \quad \begin{array}{l} \exists w' \in W, R_r(w, w') \text{ and } \langle W, (R_r)_{r \in \text{REL}}, V, S' \rangle, w' \models \varphi \\ \text{where } S' = S \cup \{w \mid \mathcal{M}, w \models p\}. \end{array}$$

In this dissertation we are going to restrict ourselves to $\langle\langle r \rangle\rangle$. We want to use it as a proof of concept to show that the interactions between modal and memory operators are a factor to consider when looking for the appropriate fragment for a given application.

2.1.3 Putting it all together

Since we are going to work with many different fragments and notations, let's sum up what we have talked about until now. We defined several memory operators, taking sets as the storage structure, so we formally introduce the syntax and semantics of (almost) all the memory logics that we will investigate. All the languages we introduce are obtained by extending the syntax and semantics of the basic modal logic.

2.1.1. Definition. [Syntax] We extend the syntax given in Definition 1.2.1 over a given signature $\langle \text{PROP}, \text{REL} \rangle$ with the following operators:

$$\varphi ::= \cdots \mid ⓚ \mid ⓡ\varphi \mid ⓔ\varphi \mid ⓕ\varphi \mid \langle\langle r \rangle\rangle \varphi$$

where $\varphi \in \text{FORMS}$ and $r \in \text{REL}$. We define the dual of $\langle\langle r \rangle\rangle$ in the usual way: for all $r \in \text{REL}$, $[\![r]\!]\varphi$ can be defined as $\neg\langle\langle r \rangle\rangle\neg\varphi$.

The models for these languages are standard Kripke models (refer to Definition 1.2.2) with the addition of a set used as a storage structure.

2.1.2. Definition. [Memory Kripke models] Given a signature $\mathcal{S} = \langle \text{PROP}, \text{REL}\rangle$, a *memory Kripke model* $\mathcal{M}$ for $\mathcal{S}$ is a tuple $\langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ satisfying the following conditions: (i) W is a nonempty set; (ii) each R_r is a binary relation on W; (iii) $V : \text{PROP} \to 2^W$ assigns to each propositional symbol $p \in \text{PROP}$ a subset of W and (iv) S, the *memory* of the model, is a set such that $S \subseteq W$.

In the rest of the thesis the following notation will be useful. Let $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ be a model and $w \in W$, then we define

$$\begin{array}{rcl} \mathcal{M}[+w] & = & \langle W, (R_r)_{r\in\text{REL}}, V, S \cup \{w\}\rangle \\ \mathcal{M}[-w] & = & \langle W, (R_r)_{r\in\text{REL}}, V, S\backslash\{w\}\rangle. \end{array}$$

For $[\circ_1 w_1, \ldots, \circ_n w_n]$ a nonempty ordered list with $w_i \in W$ and $\circ_i \in \{+, -\}$, let $\mathcal{M}[\circ_1 w_1, \ldots, \circ_n w_n] = (\mathcal{M}[\circ_1 w_1])[\circ_2 w_2, \ldots, \circ_n w_n]$, where $\mathcal{M}[\ ] = \mathcal{M}$. We will usually write $[w_1, \ldots, w_n]$ instead of $[+w_1, \ldots, +w_n]$. Additionally, given a set $S' \subseteq W$ we define

$$\begin{array}{rcl} \mathcal{M}[S'] & = & \langle W, (R_r)_{r\in\text{REL}}, V, S \cup S'\rangle \\ \mathcal{M}[-S'] & = & \langle W, (R_r)_{r\in\text{REL}}, V, S \setminus S'\rangle. \end{array}$$

In the same way as before, for $[\circ_1 S_1, \ldots, \circ_n S_n]$ a nonempty ordered list with $S_i \subseteq W$ and $\circ_i \in \{+, -\}$, let $\mathcal{M}[\circ_1 S_1, \ldots, \circ_n S_n] = (\mathcal{M}[\circ_1 S_1])[\circ_2 S_2, \ldots, \circ_n S_n]$, where $\mathcal{M}[\ ] = \mathcal{M}$. We will usually write $[S_1, \ldots, S_n]$ instead of $[+S_1, \ldots, +S_n]$.

When the context is clear enough, we will just say model instead of memory Kripke model. Let's summarize now the semantics for the operators we have introduced:

2.1.3. Definition. Given a model $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ and $w \in W$, we extend the semantics presented in Definition 1.2.4 with the following rules:

$$\begin{array}{rcl} \mathcal{M}, w \models \textcircled{k} & \text{iff} & w \in S \\ \mathcal{M}, w \models \textcircled{r}\varphi & \text{iff} & \mathcal{M}[w], w \models \varphi \\ \mathcal{M}, w \models \textcircled{f}\varphi & \text{iff} & \mathcal{M}[-w], w \models \varphi \\ \mathcal{M}, w \models \textcircled{e}\varphi & \text{iff} & \langle W, (R_r)_{r\in\text{REL}}, V, \emptyset\rangle, w \models \varphi \\ \mathcal{M}, w \models \langle\langle r\rangle\rangle\varphi & \text{iff} & \exists w' \in W, R_r(w, w') \text{ and } \mathcal{M}[w], w' \models \varphi. \end{array}$$

As we said before, the class of models where the memory is empty is particularly interesting. We denote this class as $\mathcal{C}_\emptyset = \{\mathcal{M} \mid \mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, \emptyset\rangle\}$. It is worth noting that when we say that we restrict ourselves to this class we mean that a formula is *initially* evaluated in a model of $\mathcal{C}_\emptyset$, but during its evaluation the model can change to one with a nonempty memory.

For notational convenience, let us assume fixed from now on the models $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$, $\mathcal{M}_1 = \langle W_1, (R^1_r)_{r\in\text{REL}}, V_1, S_1\rangle$ and $\mathcal{M}_2 = \langle W_2, (R^2_r)_{r\in\text{REL}}, V_2, S_2\rangle$.

Memory logics and its notational convention

We will not consider all possible combinations of operators, since it is not our intention to be completely exhaustive. We are only going to analyze some combinations that we consider interesting, and each time we will point out over which fragments we are working with. Furthermore, even though we presented the syntax and semantics of the different memory operators extending the basic modal logic, sometimes we will extend in the same way the basic hybrid language. So, let's clarify the notation we are going to use in the rest of this dissertation. We call $\mathcal{ML}$ the basic modal logic (defined in Section 1.2) and $\mathcal{HL}$ the basic hybrid logic (that is, the basic modal language augmented with nominals, see Section 1.3.3). When we add a set as a storage structure and the basic set operators ⓡ and ⓚ, we indicate this appending m as a superscript to the corresponding logic. Then we will list the additional operators included in the language. Since the usual semantics of the diamond operator is going to be slightly modified in some cases, we will also include the diamond explicitly in this list (and take the box operator as the usual shorthand). For example, $\mathcal{ML}^m(\langle r\rangle, ⓔ)$ is the modal logic with the classic diamond operator extended with ⓡ, ⓚ and ⓔ, and, to give another example, $\mathcal{HL}^m(@, \langle r\rangle)$ is the basic hybrid logics with the usual diamond, ⓡ, ⓚ and the @ operator.

In some cases we are going to restrict the class of all models, and work with $\mathcal{C}_\emptyset$, the class of models with an empty memory. To indicate that, we add $\emptyset$ as a subscript. For example, $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ is the basic modal logic with $\langle\langle r\rangle\rangle$ instead of $\langle r\rangle$, the operators ⓡ and ⓚ, and whose models have an initially empty memory.

Observe that in the notational convention we are condensing two different "dimensions". On one hand, we are specifying different *languages*, in the sense that we are adding or removing operators, and therefore changing the *syntax* of our logics. On the other hand, we are distinguishing two different classes of models: the class of all models, and the class of models with an empty memory. In this way, we are changing the *logic* itself, without necessarily modifying the syntax.

2.2 Getting to know your logic

In the previous sections we introduced changes in classical modal logics, but what is exactly the impact those changes cause in the resulting logics? The main aim of this thesis is precisely that: to analyze properties of different memory logics, in order to get to know these new members of the modal logic family.

As we mentioned in Section 1.5, we want to examine memory logics in terms of expressivity, complexity, interpolation, and proof theory. To carry out this task, we need suitable logic tools that allow us to explore these issues. Here we are going to present a basic toolkit that will help us throughout this work.

2.2.1 Expressivity

In Chapter 1 we said modal logics are not an isolated logic system. Recall that the semantics of modal operators can be thought of as performing a *guarded* quantification over elements of the domain, in contrast with the general quantification capability of classical first order logic. This makes modal logics *a priori* less expressive than their first order relative. We talked about that in an intuitive way, but how can we prove it formally?

One way to see that a logic is a fragment of another is by using *translations*, that is, a systematic way to transform a formula from one logic to the other preserving equivalence. First of all, we have to be sure that the two logics we want to compare can talk about the same kind of objects, in order to establish a common ground to make the comparison. Let's take, for example, the basic modal logic $\mathcal{ML}$ and first order logic $\mathcal{FOL}$. As we saw before, the objects that $\mathcal{ML}$ describes are Kripke models. What about $\mathcal{FOL}$? The first impression could be that $\mathcal{FOL}$ describes objects that are completely different from Kripke models. But we are not forced to talk about Kripke models using exclusively modal languages: Kripke models have everything needed to interpret classical languages too. To talk about a Kripke model $\langle W, (R_r)_{r\in\textsc{rel}}, V\rangle$ using $\mathcal{FOL}$, we can simply make use of a first order language with a binary relation symbol Q_r for every $r \in \textsc{rel}$, and a unary relation symbol P for every $p \in \textsc{prop}$. So, every time we have a Kripke model, we can think of it as a first order model: the structure is exactly the same, we are just using different languages to describe it. The language used in $\mathcal{FOL}$ to interpret modal languages is called the *first order correspondence language* for the basic modal language over PROP and REL. It is called "correspondence language" because every basic modal formula in the signature $\langle \textsc{prop}, \textsc{rel}\rangle$ corresponds to a first order formula via the *standard translation*:

$$\begin{array}{rcl}
\mathsf{ST}_x(p) & = & P(x) \quad \text{where } p \in \textsc{prop} \\
\mathsf{ST}_x(\neg\varphi) & = & \neg\mathsf{ST}_x(\varphi) \\
\mathsf{ST}_x(\varphi \wedge \psi) & = & \mathsf{ST}_x(\varphi) \wedge \mathsf{ST}_x(\psi) \\
\mathsf{ST}_x(\langle r\rangle\varphi) & = & \exists y(xQ_ry \wedge \mathsf{ST}_y(\varphi)) \quad \text{where } y \text{ is new.}
\end{array}$$

Let's analyze this translation. Propositional symbols are mapped to unary predicates, booleans are treated uniformly and diamond and boxes are handled by explicit first order quantification over Q_r accessible points. Recall that we defined the box operator as a shorthand. Applying that definition, the translation for this case is $\mathsf{ST}_x([r]\varphi) = \forall y(xQ_ry \to \mathsf{ST}_y(\varphi))$. Observe that the translation is making explicit the guarded quantification of modal logics, controlled by the relation Q_r. The variable y used in the translation for diamonds and boxes is chosen to be one that has not been used so far. Note that the formulas in the image of the translation contains exactly one free variable x. This variable, now explicit, is encoding the internal perspective of modal logics: assigning a value to this variable is equivalent to evaluating a modal formula at a particular point in

the model. But the key point is that we have defined an *equivalence preserving* translation:

2.2.1. Proposition ([BdRV01]). *Let φ be any modal formula. Then for any Kripke model $\mathcal{M}$, and any point $w \in \mathcal{M}$ we have that $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}, g_w^x \models \mathsf{ST}_x(\varphi)$.*

Here g_w^x is an arbitrary assignment function mapping first order variables to elements of the model domain, with the condition that x is mapped to w. Observe that the model $\mathcal{M}$ is a Kripke model in the left side of the equivalence, and it is a first order model in the right side. But as we said before, it is the same structure in both sides, just described with different languages.

This translation shows us that basic modal logic is indeed a fragment of first order logic: every time we have a modal formula we can translate it to a first order formula preserving equivalence. In this sense, basic modal logic is *included* in first order logic in terms of expressivity.

This translation acts also as a bridge that allow us to 'import' results from first order logic. For example, the set of validities in the basic modal language is recursively enumerable. This is easy to see using the standard translation and the fact that the set of validities for first order formulas is recursively enumerable: a modal formula φ is valid iff $\mathsf{ST}_x(\varphi)$ is valid. Other results can be transfered as well, such as *compactness* (if every finite subset of a set of formulas Γ is satisfiable, then Γ is satisfiable) and the *Löwenheim-Skolem property* (if a set of formulas Γ is satisfiable in an infinite model, then Γ is satisfiable by a model of cardinal k, for every cardinal number k).

Bisimulations

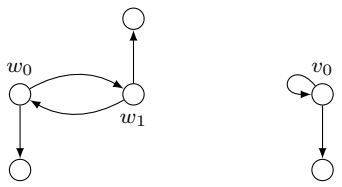
The standard translation shows us that every modal formula has an equivalent first order counterpart. But what happens in the other direction? Could it be the case that every first order formula (in the appropriate signature of course) has an equivalent modal formula? The intuition says that this is not possible, and that basic modal logic is a *proper* fragment of first order logic. How can we prove that? In other words, how can we be sure that there is *no possible translation* from first order logic to the basic modal logic?

The expressive power of a language can be measured in terms of the distinctions the language can draw. In terms of models, the question would be: when should two models be viewed as modally identical? That is, when do the same modal formulas hold in both models? Let's formalize first this notion.

2.2.2. Definition. Let $\mathcal{L}$ be a language with signature $\mathcal{S}$, $\mathcal{M}$ and $\mathcal{M}'$ two models over $\mathcal{S}$, and w and w' two points in $\mathcal{M}$ and $\mathcal{M}'$ respectively. The $\mathcal{L}$-theory of w is the set of formulas in $\mathcal{L}$ satisfied at w, that is, $\{\varphi \mid \mathcal{M}, w \models \varphi\}$. We say that w and w' are $\mathcal{L}$ equivalent (notation: $w \leftrightsquigarrow_{\mathcal{L}} w'$) if they have the same $\mathcal{L}$-theories.

When the context is clear enough, we will drop the subscript and use $\leftrightsquigarrow$ directly.

Consider now the following two models:



Let's consider now the basic modal language. Assuming that $V(p) = \emptyset$ in both models for all $p \in$ PROP, is there a way to distinguish w_0 from v_0 in $\mathcal{ML}$? That is, is there a basic modal formula that is true at w_0 and false at v_0? This doesn't seem to be easy to find. On the other hand, if we can use first order logic this is quite straightforward: the formula $\neg R(x, x)$ is true if we assign w_0 to x, and false in the case of v_0.

The idea of asking when two different structures are identical for a given language lies within a long established tradition among mathematicians of looking for the structure preserving morphisms in a given domain. The first formulation of this notion in the context of modal logic was made by van Benthem [vB84, vB85], who introduced the concept of *bisimulations*:

2.2.3. Definition. [Bisimulation] A bisimulation between two models $\mathcal{M} = \langle W, R, V\rangle$ and $\mathcal{M}' = \langle W', R', V'\rangle$ is a non-empty binary relation $E \subseteq W \times W'$ between their domains such that whenever wEw' we have that:

Atomic harmony: w and w' satisfy the same propositional symbols.

Zig: if wRv, then there exists a point v' in $\mathcal{M}'$ such that vEv' and $w'R'v'$.

Zag: if $w'R'v'$, then there exists a point v in $\mathcal{M}$ such that vEv' and wRv.

If there is a bisimulation between two models $\mathcal{M}$ and $\mathcal{M}'$ we say that $\mathcal{M}$ and $\mathcal{M}'$ are bisimilar and we write $\mathcal{M} \leftrightarrows \mathcal{M}'$. Moreover, we say that two points $w \in \mathcal{M}$ and $w' \in \mathcal{M}'$ are bisimilar if they are related by some bisimulation, and we write $\mathcal{M}, w \leftrightarrows \mathcal{M}', w'$.

One can give an equivalent notion in terms of *morphisms*, that relates two models through a *function* rather than through a relationship. In the context of modal logic they are called *bounded morphisms*.

2.2.4. Definition. [Bounded morphisms] Let $\mathcal{M} = \langle W, R, V\rangle$ and $\mathcal{M}' = \langle W', R', V'\rangle$ be two models. A mapping $f : W \to W'$ is a bounded morphism if it satisfies the following conditions:

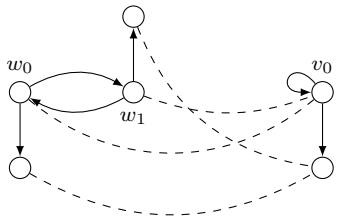
Atomic harmony: w and $f(w)$ satisfy the same propositional symbols.

Zig: if wRv, then $f(w)R'f(v)$.

Zag: if $f(w)R'v'$, then there exists a point v in $\mathcal{M}$ such that $f(v) = v'$ and wRv.

We gave the definition for the uni-modal case, but this can be easily generalized to the multi-modal case. Observe that the definitions of bisimulation and bounded morphism are quite similar: a bounded morphism is just a functional view of a bisimulation.

Returning to the models $\mathcal{M}_1$ and $\mathcal{M}_2$ we have just presented, it is easy to see that $\mathcal{M}_1, w_0 \leftrightarrows \mathcal{M}_2, v_0$. The bisimulation would be as follows (the dotted line indicates the pairs in the bisimulation relationship):

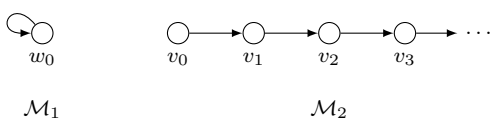


The definition of bisimulation we just gave is specifically designed for the basic modal logic, and thus the expected property is that satisfiability of formulas in the basic modal logic is invariant under bisimulations:

2.2.5. THEOREM. *[BdRV01] Let $\mathcal{M}$ and $\mathcal{M}'$ be two Kripke models over the same signature. Then, for every $w \in \mathcal{M}$ and $w' \in \mathcal{M}'$, $w \leftrightarrows w'$ implies that $w \leftrightsquigarrow w'$.*

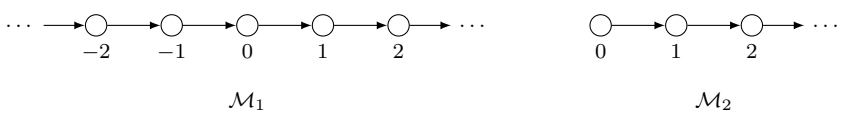
The standard translation told us that the basic modal logic is a fragment of first order logic, but we have just seen that in fact this fragment is *proper*. Why? Let's assume the opposite, and suppose there is an equivalence preserving translation Tr from first order logic to the basic modal logic. Take the first order formula $\neg R(x,x)$ (who is in the appropriate signature) and its corresponding modal formula, $\mathsf{Tr}(\neg R(x,x))$. Take again the two models $\mathcal{M}_1$ and $\mathcal{M}_2$ we presented above. Because $\mathcal{M}_1, w_0 \leftrightarrows \mathcal{M}_2, v_0$, we know that both points satisfy the same modal formulas, that is $\mathcal{M}_1, w_0 \leftrightsquigarrow \mathcal{M}_2, v_0$. In particular, this happens with the translation of $\neg R(x,x)$: $\mathcal{M}_1, w_0 \models \mathsf{Tr}(\neg R(x,x))$ iff $\mathcal{M}_2, v_0 \models \mathsf{Tr}(\neg R(x,x))$. But, as we saw before, $\mathcal{M}, g^x_{w_0} \models \neg R(x,x)$ but $\mathcal{M}', g^x_{v_0} \not\models \neg R(x,x)$, so the translation does not preserves equivalence. This implies that the translation cannot exist.

Bisimulations can also tell us that there are certain *structural* properties that the basic modal logic cannot discern. Let's see a couple of examples of this. For example, the basic modal logic cannot distinguish *cycles* in a model. What does this mean? Suppose we have these two models:



And assume also that propositional symbols do not help us to distinguish them, that is $V(p) = \emptyset$ for all $p \in \text{PROP}$ for $\mathcal{M}_1$ and $\mathcal{M}_2$. It is easy to see that the relation $(w_0, v_0), (w_0, v_1), (w_0, v_2), \ldots$ is indeed a bisimulation. In fact, $\mathcal{M}_2$ is what is called the *unraveling* of $\mathcal{M}_1$, that is, the result of a model operation that unfold the cycles of a model starting from a given point.

Another example is the *generated submodel* property. This property reflects the intrinsically local behavior of the basic modal logic. Intuitively, it says that the satisfiability of a formula at a point w is not affected by points that are not reachable from w. Let's try to clarify this. Suppose we have a model $\mathcal{M}_1$ based on the frame $(\mathbb{Z}, <)$, that is, the integers with their usual order. Suppose we form a submodel $\mathcal{M}_2$ by throwing away all the negative numbers, and restricting the valuation (whatever it was) to the remaining numbers. The two models look something like this:



Suppose a formula φ is satisfied at some point n in $\mathcal{M}_1$, with $n \geq 0$. Is φ also satisfied at the same point in $\mathcal{M}_2$? The answer to this question is *yes*, and it is quite easy to see that one can establish a bisimulation that links every point n in $\mathcal{M}_2$ with the same point in $\mathcal{M}_1$. The important thing here is to observe that given an evaluating point $n \geq 0$, the only points relevant to φ's satisfiability are the points greater than n, and all those points belong to $\mathcal{M}_2$.

This example shows that the basic modal logic cannot discern between submodels closed under the accessibility relation of the original model. Such models are called *generated submodels* (or *generated subframes*, if we are dealing with frames instead of models) and they show another invariance principle for the basic modal logic. There are also other satisfiability preserving operators, but here we will not go further in that direction. See [BdRV01] for a formal definition and details.

Bisimulations as a game

The notion of bisimulation can also be presented using a more dynamic perspective, closer to a form of process equivalence. The task of determining whether two models are bisimilar can be recast in the form of an Ehrenfeucht-Fraïssé game [EFT84].

Let $\langle \mathcal{M}_1, w_1 \rangle$ and $\langle \mathcal{M}_2, w_2 \rangle$ be two pointed models. An *Ehrenfeucht-Fraïssé game for the basic modal logic* (notation: $EF(\mathcal{M}_1, \mathcal{M}_2,\ w_1, w_2)$) is defined as follows. There are two players called *Spoiler* and *Duplicator*. The two players compare successive pairs, starting from $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$. Duplicator immediately loses if w_1 and w_2 do not coincide in the propositional symbols.

Otherwise, the game starts, with the players moving alternatively. Spoiler always makes the first move in a turn of the game, starting by choosing in which model he will make a move. In subsequents rounds, Spoiler chooses a point in one model which is a successor of the current w_1 or w_2, and Duplicator responds with a matching successor in the other model. If the chosen points differ in the atomic propositions, Spoiler wins. If one player cannot move, the other wins. Duplicator wins on infinite runs of which Spoiler does not win.

Note that with this definition, exactly one of Spoiler or Duplicator wins each game. A *strategy for Duplicator* is a function that takes an Ehrenfeucht-Fraïssé game and a sequence of valid plays of odd length and return a possible next move for Duplicator. A strategy for Spoiler is defined in the same way, substituting "odd" by "even". We say that a player is *following a strategy* s when all his moves in a game comply with the answer of s for every stage of the game. A strategy is *winning* if the player following it necessarily wins the game, no matter what his opponent plays. Given two pointed models $\langle \mathcal{M}_1, w_1 \rangle$ and $\langle \mathcal{M}_2, w_2 \rangle$ we will write $\langle \mathcal{M}_1, w_1 \rangle \equiv^{EF} \langle \mathcal{M}_2, w_2 \rangle$ when Duplicator has a winning strategy for $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$.

Intuitively, this game captures exactly the zigzag behavior of bisimulations, and the atomic harmony condition. The two notions are equivalent, but depending on the context, one can be more natural than the other.

2.2.6. Proposition. *[GO05] Given two pointed models $\langle \mathcal{M}_1, w_1 \rangle$ and $\langle \mathcal{M}_2, w_2 \rangle$ then $\langle \mathcal{M}_1, w_1 \rangle \equiv^{EF} \langle \mathcal{M}_2, w_2 \rangle$ if and only if $\langle \mathcal{M}_1, w_1 \rangle \leftrightarrows \langle \mathcal{M}_2, w_2 \rangle$.*

Summing up, bisimulations are a very powerful tool to measure the expressivity of a logic: it provides us with *structural* conditions on the models that characterizes the appropriate structure preserving morphisms. Since bisimulations are directly linked to the expressivity of a given logic, there is not a *unique* notion of bisimulations. Here we have just presented the bisimulation for the basic modal logic, but for *every* logic we need to find a suitable definition of bisimulation, and this notion is a direct reflex of the logic expressive power. In this sense, looking for the appropriate bisimulation allow us to *learn* about the logic we are working with. In the next chapter we are going to see the associated bisimulations for some of the fragments of the memory logic family, and we will use these notions to compare them with other well known modal and hybrid logics.

2.2.2 Decidability

We are also going to examine the computability of the satisfiability problem for some members of the memory logic family. The abstract formulation of the question would be: given a modal formula φ and a class of models $\mathcal{C}$, is it computable whether φ is satisfiable in a model of $\mathcal{C}$?

There are several techniques to try to determine this matter. Here we will present just a few of them that will help us later deal with our particular memory fragments. From now on, we call a logic *decidable* if its satisfiability problem is decidable.

Decidability via finite models

The path to choose to establish decidability depends much on the way we 'know' the logic we are working with. We may now a logic $\mathcal{L}$ purely semantically: we have only the class of models and its satisfaction relationship. On the other hand, we may also have a syntactic definition: $\mathcal{L}$ is the logic generated by some set of axioms. In both cases, establishing that a logic has the finite model property is a useful first step for proving decidability.

2.2.7. Definition. A model is *finite* if it has a finite domain, finitely many nonempty relations, and a valuation function that assigns nonempty sets to finitely many propositional symbols. $\mathcal{L}$ has the *finite model property* (f.m.p) if every time a formula φ of $\mathcal{L}$ is satisfiable, then it is satisfiable in a finite model.

In this context, there are two possible strategies for establishing decidability. We just give an informal argument for them, to see a rigorous analysis refer to [BdRV01]. Suppose we only have a semantic specification of $\mathcal{L}$, but we have been able to prove that $\mathcal{L}$ has the finite model property. Even more, we have been able to prove that for any formula $\varphi \in \mathcal{L}$ there is a computable function f such that $f(\varphi)$ is an upper bound on the size of the model that could satisfy φ (this is usually called the *strong* or *bounded* finite model property). In this scenario we can decide whether a formula φ is satisfiable as follows: we write a Turing machine that takes φ as input, generates all the finite models up to size $f(\varphi)$, and tests for the satisfiability of φ on these models. Because the logic has the strong finite model property, and the Turing machine explores all the models up to the appropriate size, the machine decides the satisfiability of φ.

On the other hand, suppose $\mathcal{L}$ is given through a sound and complete axiomatization (assume also that the set of axioms and rules is recursively enumerable), and we have been able to prove that $\mathcal{L}$ has the finite model property. First we construct a Turing machine that uses the axiomatization to recursively enumerate the theorems of $\mathcal{L}$. Second, we construct another Turing machine that recursively enumerates all the finite models. So now we can start the two machines at the same time: if the formula is a $\mathcal{L}$-validity, the first machine will eventually generates φ and stops. If that is not the case, the second machine will eventually falsify φ on a finite model. Thus, we can decide in this way the validity of φ (and therefore the satisfiability of $\neg\varphi$).

We described these two procedures to work with the class of all models of a given logic. If we are dealing with the logic of a specific class of models $\mathcal{C}$, observe

that we also need to have a computable procedure to identify if a given finite model $\mathcal{M}$ belongs to $\mathcal{C}$.

Undecidability via tiling

A way to prove that the satisfiability problem of a logic $\mathcal{L}$ is undecidable is to reduce it to some known undecidable problem. There are many candidates for making the reduction, but the *tiling problem* is particularly suitable for modal logic.

A tiling problem is essentially a jigsaw puzzle made of *tiles*. A tile T is a square, each side of which has a color. The orientation of each tile is fixed, that is, we cannot rotate a tile. In the following picture we show some tiles:

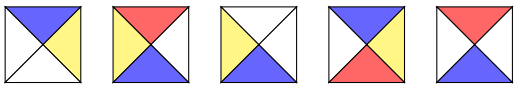


Figure 2.5: Some different tile types

A simple tiling problem would be: is it possible to arrange tiles of the types shown in Figure 2.5 on a 2×3 grid such that every pair of adjacent tiles have the same color on the common side? The answer to this question is *yes*:

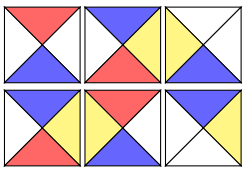


Figure 2.6: A 2×3 tiling

The idea of solving a tiling gives rise to a family of problems which can be used to analyze computational complexity and undecidability. Some tiling problems vary the type of space to be tiled, or they add some additional constraints on what counts as a successful tiling. We are going to mention just one type of tiling, which we will use to prove undecidability for many of the memory logics fragments. More details about tilings can be found in [vEB97]. The $\mathbb{N} \times \mathbb{N}$ *tiling problem* is the following:

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

We have shown an example of a tiling of 2×3, but the general case for $\mathbb{N} \times \mathbb{N}$ is hard, and in fact the following theorem holds:

2.2.9. Theorem. *[vEB97] The problem of determining the answer for the* $\mathbb{N} \times \mathbb{N}$ *tiling problem is undecidable.*

How can we use the tiling problem to prove the undecidability of a given logic $\mathcal{L}$? Given a finite set of tiles $\mathcal{T}$, we can try to construct a formula $\varphi_{\mathcal{T}} \in \mathcal{L}$ such that

$$\mathcal{T} \text{ tiles } \mathbb{N} \times \mathbb{N} \text{ iff } \varphi_{\mathcal{T}} \text{ is satisfiable.}$$

If we can construct such a formula, then the satisfiability problem for $\mathcal{L}$ is undecidable. Why? Suppose that it was decidable. Then we could solve the $\mathbb{N} \times \mathbb{N}$ tiling problem as follows: given $\mathcal{T}$, we build $\varphi_{\mathcal{T}}$ and use the decision procedure for $\mathcal{L}$ to decide whether $\varphi_{\mathcal{T}}$ is satisfiable. This would solve the tiling problem, but that is impossible given Theorem 2.2.9.

2.2.3 Interpolation and Beth definability

Interpolation and Beth definability are properties that we are also going to study in the context of memory logics. We say that a modal logic has *interpolation over propositional symbols* on a frame class $\mathcal{C}$, if for all formulas φ, ψ such that $\mathcal{C} \models \varphi \to \psi$, there is a modal formula δ (usually called the *interpolant*) such that $\mathcal{C} \models \varphi \to \delta$ and $\mathcal{C} \models \delta \to \psi$, and $\mathsf{props}(\delta) \subseteq \mathsf{props}(\varphi) \cap \mathsf{props}(\psi)$. Note that there is no restriction on the modalities occurring in δ.

Why do we want to study interpolation? One answer is that interpolation is important as a modularity principle. Suppose there are two system specifications, represented as sets of formulas Σ and Γ, and these specifications contradict each other. If we have compactness, then the interpolation property tells us that there is a formula φ in the common language, on which Σ and Γ disagree. In other words, there is going to be a witness of the contradiction expressed in terms of a set of symbols that both logics talk about. Other reason is that interpolation has been considered an indicator of the existence of a cut-free sequent calculi for the logic in question [BFB85]. Also interpolation is often useful to prove the Beth property, which we will discuss later.

Observe that in the setting of hybrid logics, there is a choice to make. Do we want to include the nominals in the common language? The first choice is to require that the interpolant of a valid implication must contain only proposition letters occurring both in the antecedent and in the consequent. No restriction is made on the occurrence of nominals in the interpolant. The other option would be to require that both the proposition letters and the nominals occurring in the interpolant occur both in the antecedent and the consequent. We will refer to these options as *interpolation over proposition letters* and *interpolation over proposition letters and nominals* respectively.

In this work we are going to study a type of interpolation (that is the one we just have defined above) called *local interpolation* or *arrow interpolation*. There is another type of interpolation called *global* or *turnstile* interpolation, that we will not analyze. More information about the different types of interpolation and motivations for studying interpolation in general can be found in Hoogland's

dissertation [Hoo01].

Let's talk now about the Beth property. First we need to introduce the concept of global entailment. Recall that in Section 1.2 we introduced just the local version of semantic consequence. A formula φ is a *global semantic consequence* of a set of formulas Σ (notation: $\Sigma \models^{glo} \varphi$) if for all models $\mathcal{M}$, if $\mathcal{M} \models \Sigma$, then $\mathcal{M} \models \varphi$. Clearly, local semantic consequence implies global semantic consequence. For a set of formulas $\Sigma(p)$ containing the proposition letter p (and possibly other proposition letters), we say that $\Sigma(p)$ *implicitly defines* p, relative to a frame class $\mathcal{C}$, if $\Sigma(p) \cup \Sigma(p') \models^{glo}_{\mathcal{C}} p \leftrightarrow p'$. Here, p' is a propositional symbol not occurring in Σ, and $\Sigma(p')$ is the result of replacing all occurrences of p by p' in $\Sigma(p)$. A modal logic is said to have the *Beth property* relative to a frame class $\mathcal{C}$, if whenever a set of modal formulas $\Sigma(p)$ implicitly defines a proposition letter p, then there is a modal formula δ in which p does not occur, such that $\Sigma \models^{glo}_{\mathcal{C}} p \leftrightarrow \delta$. The formula δ is called an *explicit definition* of p relative to Σ and $\mathcal{C}$.

These two properties are very much related, and usually they are both present at the same time in a given logic (it is not easy to find a logic where one property is present and the other is absent, refer to [Hoo01] for details). Observe also that the definition of interpolation uses the notion of local semantic consequence, while Beth definability uses global consequence. Intuitively, the notion of Beth involves the synchronization between implicit and explicit definition, and implicit definition involves a set of formulas $\Sigma(p)$ on the *left* side of the turnstile. That makes $\Sigma(p)$ impose conditions on the models in a global way, since it does not bound any specific point in the model. On the other hand, interpolation involves being able to find an interpolant for $\varphi \to \psi$, where this formula holds in a local sense, at the *right* side of the turnstile: *at every given point* where φ holds, then ψ must also hold.

Intuitively, if a logic has the Beth property this can be interpreted as a sign that its syntax and semantics match well. Both interpolation and Beth definability are strongly connected to the expressivity of a logic. In Chapter 5 we are going to see that many of the results we present in Chapter 3 are going to be 'reused', but now in connection with these properties.

2.2.4 Axiomatizations

Until now we have been mainly talking about *model theoretical* issues, but logic has a substantial *proof theoretical* perspective. In this dissertation we are going to focus in Hilbert-style and tableau approaches to modal proof theory in the context of memory logics. We are going to discuss semantic tableau in the next section.

The general question related to the syntactic side of a logic $\mathcal{L}$ is: are there syntactic mechanisms capable of generating all the valid formulas of $\mathcal{L}$? We present here a Hilbert-style axiom system called **K**. This axiomatic system can be thought of as a system for reasoning about the class of all Kripke models.

Stronger systems can be obtained by adding extra axioms. It is not a coincidence then that **K** is the system corresponding to the basic modal logic. Let's define first some basic concepts before we explain this in more detail.

2.2.10. Definition. Given an axiomatization $\mathcal{A}$, an $\mathcal{A}$-*proof* is a finite sequence of formulas, each of which is an *axiom* of $\mathcal{A}$, or follows from one or more earlier items in the sequence by applying a *rule of proof* of $\mathcal{A}$. Any element of the sequence of a proof is called a *theorem* of $\mathcal{A}$. We write $T(\mathcal{A})$ for the set of all theorems in $\mathcal{A}$.

So let's present now the axiomatic system **K**, and try to understand it from an intuitive point of view:

Axioms:	
CT	All instances of propositional tautologies
K	$\vdash [r](p \to q) \to ([r]p \to [r]q)$
Rules:	
MP	If $\vdash \varphi$ and $\vdash \varphi \to \psi$ then $\vdash \psi$
Gen	If $\vdash \varphi$ then $\vdash [r]\varphi$
Sub	If $\vdash \varphi$ then $\vdash \varphi[p/\psi]$ for any $p \in$ PROP

Figure 2.7: Axiomatization for $\mathcal{ML}$.

The expression $\varphi[a/b]$ is the result of uniformly replacing all occurrences of a in φ by b. Note that instances of propositional tautologies will contain modalities, for example, $\langle r\rangle p \vee \neg\langle r\rangle p$ is an instance of a propositional tautology, as it has the same form as $p \vee \neg p$. Modus ponens (*MP*) is probably very familiar to everybody, so we won't go into details here. The uniform substitution rule (*Sub*) reflects the fact that validity does not depend on a particular assignment: if a formula is valid, this is not because of a particular assignment on its propositional symbols, so we should be able to replace these symbols with any other symbols. Let's turn now to the genuinely *modal* components of the system. The axiom K is often called the *distribution axiom*, and together with MP, let us transform a boxed formula $[r](\varphi \to \psi)$ in an implication $[r]\varphi \to [r]\psi$. The ability to distribute the box over the implication allows further propositional reasoning to take place. Finally, the *generalization* rule (*Gen*) enable us to stack boxes in front of validities. While the axiom K let us apply purely propositional reasoning inside boxed formulas, the generalization rule creates new modal contexts to work with.

We said before that the system **K** *is* the axiomatic system for the basic modal logic. What is the exact meaning of that? In a nutshell, that the theorems generated by **K** and the validities derived from the semantic definition are *exactly* the same set of formulas. That is to say, that the syntactic and the semantic definition perfectly match. Let's define this concepts formally.

2.2.11. Definition. [Consistency, Soundness, Completeness] We say that a formula φ is *consistent* with respect to an axiomatization $\mathcal{A}$ (or $\mathcal{A}$-consistent) if $\neg\varphi$ is not a theorem of $\mathcal{A}$. The notion of consistency can be extended to a set of formulas Γ by requiring that for no finite subset Γ^f, the formula $\bigwedge \Gamma^f \to \neg\top$ be a theorem of $\mathcal{A}$.

Recall that given a class of models $\mathcal{C}$, we denote $T(\mathcal{C})$ the set of all valid formulas in $\mathcal{C}$. Given an axiomatization $\mathcal{A}$ and a class of models $\mathcal{C}$ we say that $\mathcal{A}$ is *sound for* $\mathcal{C}$ if $T(\mathcal{A}) \subseteq T(\mathcal{C})$, and that it is *complete for* $\mathcal{C}$ if $T(\mathcal{C}) \subseteq T(\mathcal{A})$. Completeness can be equivalently defined in terms of consistency and satisfiability: $\mathcal{A}$ is complete for $\mathcal{C}$ if every formula consistent in $\mathcal{A}$ is satisfiable in $\mathcal{C}$.

Finally, we say that an axiomatization $\mathcal{A}$ is *strongly complete* with respect to $\mathcal{C}$, if every $\mathcal{A}$-consistent set of formulas is satisfiable in $\mathcal{C}$.

Reformulating what we said before with these new definitions, we can state the following theorem:

2.2.12. Theorem. *[BdRV01] **K** is sound and strongly complete with respect to the class of all models.*

We are interested in providing sound and complete axiomatizations for memory logics, and since memory logics are an extension of the basic modal logic, the axiomatic systems we are going to give are extensions of the system **K**. As it will be clear from the details that we present in Chapter 6, nominals and @ will play a crucial role in these axiomatic characterizations.

2.2.5 Tableau calculus

Axiomatic systems can be regarded as *forward reasoning* systems, in the sense that one starts with axioms and rules, and finishes with the desired theorem. They show in a very clear way the *interaction* among different logic operators, and many of them are very elegant systems. As we have seen, the basic modal logic can be described just by extending a propositional axiomatic system with the **K** axiom and the Generalization rule. On the other hand, axiomatic systems are often badly suited to proof discovery, and they are not usually good candidates for automated deduction. In contrast, *semantic tableau* (or *tableau* for short) is a *backward reasoning* system: it begins with the desired result and works backward from there to create a proof. Tableaus were introduced in [Bet65], and further developed in [Smu95]. They have shown to be a very flexible method for a rich variety of logics, with many successful implementations. There are other backward reasoning system (one can mention resolution-based calculus, natural deduction, Gentzen systems, sequent calculus, etc. [RV01]), but in this thesis we are going to restrict ourselves to tableau calculus. We are just going to present the tableau method from an intuitive point of view without giving full details. For further references see [DGHP99].

As a warming up for Chapter 7, we present a tableau calculus for the basic modal logic $\mathcal{ML}$. This result was originally presented in [Fit72]. To simplify the presentation of the tableau rules, we are going to work with formulas in *negation normal form* (NNF). A formula is in negation normal form if negation occurs only immediately in front of atomic symbols. For the case of the basic modal logic, each formula can be brought into this form by using standard De Morgan's laws to push negation inside propositional connectives, eliminating double negations and applying the rewriting rules $\neg\langle r\rangle\varphi \equiv [r]\neg\varphi$ and $\neg[r]\varphi \equiv \langle r\rangle\neg\varphi$. Just for the sake of convenience, we are going to work with $\mathcal{ML}$ fixing the language to the propositional connectives $\vee, \wedge$ and $\neg$, plus the modal operators $\langle r\rangle$ and $[r]$. We are also going to assume that the formulas we work with are already in NNF.

$$(\wedge)\quad \frac{w{:}\varphi \wedge \psi}{\begin{array}{c} w{:}\varphi \\ w{:}\psi \end{array}} \qquad\qquad (\vee)\quad \frac{w{:}\varphi \wedge \psi}{w{:}\varphi \mid w{:}\psi}$$

$$(\langle r\rangle)\quad \frac{w{:}\langle r\rangle\varphi}{\begin{array}{c} v{:}\varphi \\ wR_r v \end{array}}\ \dagger \qquad\qquad ([r])\quad \frac{w{:}[r]\varphi \quad wR_r v}{v{:}\varphi}$$

$$(\bot)\quad \frac{w{:}p \quad w{:}\neg p}{v{:}\bot}$$

Key: $\dagger$ v is fresh.

Figure 2.8: A prefixed tableau for the basic modal logic

In Figure 2.8 we present a *prefixed* tableau calculus for $\mathcal{ML}$. That the calculus is prefixed means that the formulas occurring in the tableau rules are *prefixed formulas* of the form $\tau{:}\varphi$ where φ is a formula and τ belongs to some fixed countably infinite set of symbols called *prefixes*. The intended interpretation of a prefixed formula $\tau{:}\varphi$ is that τ denotes a point at which φ holds. In addition to prefixed formulas, the tableau rules contain *accessibility formulas* of the form $\tau_1 R_r \tau_2$, where τ_1 and τ_2 are prefixes. The intended meaning for $\tau_1 R_r \tau_2$ is that the point denoted by τ_1 is related through R_r with the point denoted by τ_2. Observe that prefixes are not part of the object language, they are just a way to "guide" the construction of the tableau. Formally,

2.2.13. Definition. Let $W = \{w_1, w_2, \ldots, \}$ be an infinite enumerable set of labels. Then $w{:}\varphi$ is a *prefixed formula*, where $\varphi \in \mathcal{ML}$ and $w \in W$. $wR_r w'$ is an *accessibility formula* for $r \in$ REL, and $w, w' \in W$. In this context, we will use the term *formula* to denote either a formula of $\mathcal{ML}$, a prefixed formula, or an accessibility formula.

The rules are presented in the standard format: each rule has a name on the left and is divided in an upper (the antecedent) and lower (the consequent) part. Whenever there are formulas in a branch that match the antecedent, the rule can be applied following the constraints specified for each rule. If the rule is applied, the formulas of the consequent are added to the same branch, except in the case of ($\vee$), where two different branches are created.

Let's analyze the rules from an intuitive point of view. Having in mind that each branch of the tableaux represent an attempt to satisfy the root formula, the propositional rules ($\wedge$) are ($\vee$) are self-explanatory. Consider the ($\langle r \rangle$) rule. This rule decomposes the *existencial* request made by the $\langle r \rangle$ operator in two: first, there should be a w-successor point v and, second, the successor point v must satisfy φ. Since there should be no *a priori* additional constraints set on the successor of w, the prefix introduced by this rule is requested to be new. Let's analyze now the ($[r]$) rule. In essence, this rule treats formulas of the form $w{:}[r]\varphi$ as constraints on all the successors of the point labeled by w. Every time this type of formula is present and there is a prefix wRv, this rule imposes φ on the point labeled by v. Finally, the ($\perp$) rule is a typical *clash* rule. A clash is an indicator that the current attempt to satisfy the root formula is not possible. When a propositional symbol p and its negation $\neg p$ are found in the same branch, holding at the same point, this rules inject a $\perp$ in the branch, to prevent further extensions.

As is the case also for axiomatic systems, the ultimate aim of a tableau calculus is to characterize the set of valid formulas of a given logic. What is the meaning of that in terms of a tableau? Let's define that formally.

2.2.14. Definition. A *saturated tableau* is a tableau in which no new formulas can be added by applying the rules. A *saturated branch* is a branch of a saturated tableau. A branch of a tableau is called *closed* if it contains $\perp$. Otherwise the branch is called *open*. A *closed tableau* is one in which all branches are closed, and an *open tableau* is one in which at least one branch is open.

We call a tableau calculus $\mathcal{T}$ *sound* for a language $\mathcal{L}$ respect to a class of models $\mathcal{C}$ if whenever $\varphi \in \mathcal{L}$ is $\mathcal{C}$-satisfiable, then every saturated tableau T with root φ has an open branch. We say that it is *complete* if whenever $\varphi \in \mathcal{L}$ is not $\mathcal{C}$-satisfiable, then every saturated tableau T with root φ is closed.

We said that the tableau calculus we presented characterizes $\mathcal{ML}$ in the same way the Hilbert-style axiomatic system **K** does. Now we can restate that formally as follows.

2.2.15. Theorem ([Fit83]). *The tableau calculus presented in Figure 2.8 is sound and complete for $\mathcal{ML}$ with respect to the class of all models. More precisely, given $\varphi \in \mathcal{ML}$, φ is satisfiable iff any saturated tableau for $\mathcal{ML}$ with root $w{:}\varphi$ has an open branch.*

Using that φ is satisfiable if and only if $\neg\varphi$ is not valid, we can use this method to characterize both satisfiability and validity.

As we said before, tableaus showed to be a nice method to perform automated deduction. Given that the basic modal logic is decidable, one expects the tableau we presented to be a *terminating* tableau calculus, that is to say, that every tableau derivation is finite. Since the tableau rules are all finitely branching, by König lemma it suffices to see that the construction of every branch terminates, or in other words, that every saturated branch has finite length.

Observe that the tableau calculus we presented does not necessarily terminate. One reason is quite obvious: we have to restrict adding the same formula twice in the same branch. But there is a more subtle reason. Note that the $(\langle r\rangle)$ rule can be applied infinitely many times, generating a new prefix in every application. To avoid that, we are going to impose two constraints on the construction of a tableau:

1. A formula is never added to a tableau branch where it already occurs.
2. The rule $\langle r\rangle$ is never applied twice to a formula on a given branch.

Of course, these restrictions do not affect soundness. But it can also be shown (see [Fit83]) that the resulting tableau is terminating and complete.

Returning to the memory logic family, we are interested in providing sound and complete tableau calculus for some of its members. In Chapter 7 we are going to present these results. We will see that the use of *prefixed* tableaus is an effective method to gain a finer-grained control on tableau derivations. This will help to overcome some difficulties, like the ones we found with Hilbert-style systems and the lack of nominals.

2.3 How memory logics were born?

We finish this chapter giving a brief historical recap about how memory logics were conceived.

Memory logics where initially defined for purely theoretical reasons (related to questions concerning binding and decidability), but it soon became clear that they could provide an interesting perspective on the question of how can a formula modify the model in which the formula is being evaluated, as we discussed above.

They were inspired by hybrid logics like $\mathcal{HL}(\downarrow)$, but while the $\downarrow$ operator was introduced to investigate *dynamic naming* of elements in a model, memory logics include operators that let us *store* and *retrieve* information from some kind of information structure or memory. But as we will see, properly viewed, $\mathcal{HL}(\downarrow)$ could be considered the first memory logic.

As we saw in Section 1.3.3, one way of looking at the semantic condition for $\downarrow i.\varphi$ is that it dynamically creates a name for the current point (by linking the

nominal i to it), so that we can later refer to it during the evaluation of φ. An alternative perspective is to see $\downarrow i$ as an instruction to modify the model (by storing the current point of evaluation into i), and continue the evaluation of φ in the modified model. The difference between the two perspectives is subtle, but important in this context. In the latter, we are considering the assignment g as a kind of memory in our model, while $\downarrow i$ and i are the tools we use to access the memory for reading and writing. The question then presents itself naturally: are there other kinds of interesting memory structures and memory operators?

We could say that the assignment g is a very sophisticated memory structure: it has unbounded size, it provides direct access to all its memory cells, and each stored element can be unequivocally retrieved. The set S we discussed above, together with the operators ⓡ and ⓚ, provides a much simpler memory structure. Intuitively, these operators cannot discern between different points stored in S, while an assignment g keeps a complete mapping between points and nominals.

On the other hand, we can turn things around and think of ⓡ *as a binder* that effectively binds instances of ⓚ appearing in its scope. In other words, as we can see $\downarrow i$ and nominals as memory operators which store and retrieve information from a memory structure, we could see ⓡ as a binder that binds occurrences of ⓚ in its scope. Because the memory structure used by ⓡ and ⓚ has smaller discerning power, we would expect that the new operators are less expressive than $\mathcal{HL}(\downarrow)$.

Actually, as we are going to see in the next chapters, with ⓕ, ⓔ, ⓡ and ⓚ we can express properties similarly as how it is done using binders in different hybrid languages [AtC06, Bla00].

Chapter 3

Expressiveness

No sé distinguir entre besos y raíces
no sé distinguir lo complicado de lo simple

"Las chispa adecuada", Héroes del Silencio.

In this chapter we are going to formally measure the expressive power of several memory logic fragments. To be able to do that, we will investigate the notion of model equivalence and develop appropriate notions of bisimulation (both the structural and the game versions). Then we will use these tools to compare different memory fragments among them, and to examine them in contrast with $\mathcal{ML}$ and $\mathcal{HL}(\downarrow)$.

As we said before, it is not our intention to be completely exhaustive and explore all possible combinations of memory operators. We just show the comparison between some fragments that we consider interesting for exposing the impact of each different 'memory ingredient'. In many cases, the results shown for some fragments can be easily transfered to other fragments, not explicitly analyzed.

3.1 Model equivalence

Now we want to extend the notion of bisimulation we gave in Section 2.2.3 to include memory operators. In this context we also need to take care of the memory, so bisimulations will not simply link points but pairs (S, m), where S is the current memory and w a point. We start by defining the notion of bisimulation for the simplest memory logic $\mathcal{ML}^m(\langle r \rangle)$, that is, the basic modal logic extended with ⓡ and ⓚ.

Given two models $\mathcal{M}_1$ and $\mathcal{M}_2$, and points $w_1 \in W_1$ and $w_2 \in W_2$, we say that they *agree* when $\mathsf{props}(w_1) = \mathsf{props}(w_2)$ and $w_1 \in S_1$ iff $w_2 \in S_2$.

3.1.1. Definition. [Memory bisimulation] Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two models. Let $\sim$ be a binary relation between $2^{W_1} \times W_1$ and $2^{W_2} \times W_2$. So $\sim$ relates tuples $\langle \{m_1, m_2, \dots\}, m \rangle$ with $\langle \{n_1, n_2, \dots\}, n \rangle$. We write these tuples as $\langle S, m \rangle$. The bisimulation relationship should satisfy the following properties:

(*nontriv*)	$\sim$ is not empty.
(*agree*)	If $\langle S, m\rangle \sim \langle S', n\rangle$, then m and n agree.
(*zig*)	If $\langle S, m\rangle \sim \langle S', n\rangle$ and $R^1_r(m, m')$, then there exists $n' \in W_2$ such that $R^2_r(n, n')$ and $\langle S, m'\rangle \sim \langle S', n'\rangle$.
(*zag*)	If $\langle S, m\rangle \sim \langle S', n\rangle$ and $R^2_r(n, n')$, then there exists $m' \in W_1$ such that $R^1_r(m, m')$ and $\langle S, m'\rangle \sim \langle S', n'\rangle$.
(*remember*)	If $\langle S, m\rangle \sim \langle S', n\rangle$, then $\langle S \cup \{m\}, m\rangle \sim \langle S' \cup \{n\}, n\rangle$.

It is worth making some remarks here. First note that this definition extends the one given in Definition 2.2.3. The condition *(agree)* extends atomic harmony to include the memory, and therefore shows that now we have the ability to check whether a point is memorized using ⓚ. The conditions *(zig)* and *(zag)* are exactly the same conditions than for the basic modal case. The new condition *(remember)* reflects the fact that now we can add points to the current memory using ⓡ. Second, observe the *modular character* of the definition of bisimulation, in which each operator in the logic has a clause (or set of clauses) directly associated to it. That means that in fact we have defined conditions that *any* definition of bisimulation for a memory logic including the operators $\langle r\rangle$, ⓡ and ⓚ (and the standard boolean operators) must fulfill.

We can exploit this modular character, and define the notion of bisimulation for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. Since in this logic the only change involves replacing $\langle r\rangle$ for $\langle\langle r\rangle\rangle$, then resulting definition behaves as expected: we should change only the *(zig)* and *(zag)* rules, and replace them with the following two conditions:

(*mzig*)	If $\langle S, m\rangle \sim \langle S', n\rangle$ and $R^1_r(m, m')$, then there exists $n' \in W_2$ such that $R^2_r(n, n')$ and $\langle S \cup \{m\}, m'\rangle \sim \langle S' \cup \{n\}, n'\rangle$.
(*mzag*)	If $\langle S, m\rangle \sim \langle S', n\rangle$ and $R^2_r(n, n')$, then there exists $m' \in W_1$ such that $R^1_r(m, m')$ and $\langle S \cup \{m\}, m'\rangle \sim \langle S' \cup \{n\}, n'\rangle$.

In the same way as before, not only we defined a bisimulation for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, but a set of conditions that any full Boolean memory logic including the operators $\langle\langle r\rangle\rangle$, ⓡ and ⓚ must fulfill. We only have left outside the operators ⓔ and ⓕ, so let's define the associated bisimulation conditions for each of them now:

(*erase*)	If $\langle S, m\rangle \sim \langle S', n\rangle$, then $\langle \emptyset, m\rangle \sim \langle \emptyset, n\rangle$
(*forget*)	If $\langle S, m\rangle \sim \langle S', n\rangle$, then $\langle S \setminus \{m\}, m\rangle \sim \langle S' \setminus \{n\}, n\rangle$.

With these definitions, we have presented the bisimulation notions for all the fragments introduced in Definition 2.1.1. Every time we pick a specific memory fragment, we need to select the associated clauses from the definition of bisimulation to obtain the appropriate notion for it.

In the same manner we did with the bisimulation for the basic modal logic, given two pointed models $\langle \mathcal{M}_1, w_1\rangle$ and $\langle \mathcal{M}_2, w_2\rangle$ we write $\langle \mathcal{M}_1, w_1\rangle \leftrightarrows \langle \mathcal{M}_2, w_2\rangle$ if there is a bisimulation linking $\langle S_1, w_1\rangle$ and $\langle S_2, w_2\rangle$. The exact type of

bisimulation involved will usually be clear from the context, and we will write $\leftrightarrows_{\mathcal{L}}$ when we need to specify that the bisimulation corresponds to the logic $\mathcal{L}$.

In Section 2.2.1 we said that an equivalent notion of bisimulations can be presented from the perspective of games. In many contexts this alternative notion will be very useful for our purposes, so we present here the Ehrenfeucht-Fraïssé games for the fragments we are going to study. In the same way as with bisimulations, the possible moves for both players can also be presented in a modular way. Each type of move is associated with an operator, and picking a concrete set of operators automatically defines the set of possible moves in the game. Moreover, observe that the following definition extends the one we gave for the basic modal logic.

3.1.2. Definition. [Memory Ehrenfeucht-Fraïssé Games] Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two models and let $w_1 \in W_1$ and $w_2 \in W_2$.

An *memory Ehrenfeucht-Fraïssé game* $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ is defined as follows. There are two players called *Spoiler* and *Duplicator*. Duplicator immediately loses if w_1 and w_2 do not agree. Otherwise, the game starts, with the players moving alternatively. Spoiler always makes the first move in a turn of the game, starting by choosing in which model he will make a move. Let us set $s = 1$ and $d = 2$ in case he chooses $\mathcal{M}_1$; otherwise, let $s = 2$ and $d = 1$.

For the fragments including the operators $\{\langle r\rangle, ⓡ, ⓚ\}$ the possible moves are as follows:

1. Make a *memorizing step*. I.e., Spoiler extends S_s to $S_s \cup \{w_s\}$. The next turn then starts with $EF(\mathcal{M}_1[w_1], \mathcal{M}_2[w_2], w_1, w_2)$ (Duplicator does nothing in this case).

2. Make a *move step*. I.e., Spoiler chooses $r \in \text{REL}$, and v_s, an R_r^s-successor of w_s. If w_s has no R_r^s-successors, then Duplicator wins. Duplicator has to chose v_d, an R_r^d-successor of w_d, such that v_s and v_d agree. If there is no such successor, Spoiler wins. Otherwise the game continues with $EF(\mathcal{M}_1, \mathcal{M}_2, v_1, v_2)$.

The moves for the fragments that have the $\langle\langle r\rangle\rangle$ operator instead of $\langle r\rangle$ are similar, except that during a *move step* Spoiler always remembers the current world, i.e., the game goes on with $EF(\mathcal{M}_1[w_1], \mathcal{M}_2[w_2], v_1, v_2)$ after Duplicator response.

If we want to include the operators ⓔ and ⓕ, we should add the corresponding moves for the players. The associated moves for them are the following:

- Make a *erasing step*. I.e., Spoiler wipes out S_s, that is, sets $S_s = \emptyset$. The next turn then starts with $EF(\langle W_1, (R_r^1)_{r\in\text{REL}}, V_1, \emptyset\rangle, \langle W_2, (R_r^2)_{r\in\text{REL}}, V_2, \emptyset\rangle, w_1, w_2)$.

- Make a *forgetting step*. I.e., Spoiler deletes w_s from S_s, that is, sets $S_s = S_s \setminus \{w_s\}$. The next turn then starts with $EF(\langle W_1, (R^1_r)_{r\in\text{REL}}, V_1, S_1 \setminus \{w_1\}\rangle, \langle W_2, (R^2_r)_{r\in\text{REL}}, V_2,\ S_2 \setminus \{w_2\}\rangle, w_1, w_2)$.

In the same way as for the case for basic modal logic, in an infinite game Duplicator wins, and therefore exactly one of Spoiler or Duplicator wins each game. Also following the same notation we already gave, given two pointed models $\langle \mathcal{M}_1, w_1\rangle$ and $\langle \mathcal{M}_2, w_2\rangle$ we will write $\langle \mathcal{M}_1, w_1\rangle \equiv^{EF} \langle \mathcal{M}_2, w_2\rangle$ when Duplicator has a winning strategy for $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ (the exact type of game involved will usually be clear from the context, and we will write $\equiv^{EF}_{\mathcal{L}}$ when we need to specify that the game corresponds to the logic $\mathcal{L}$).

As we said before, the notions of Ehrenfeucht-Fraïssé games and bisimulations coincide, as indicated in the following theorem. We just prove it for $\mathcal{ML}^m(\langle r\rangle)$, since the cases when we have $\langle\langle r\rangle\rangle$, ⓔ or ⓕ are quite similar.

3.1.3. Theorem. *Let $\langle \mathcal{M}_1, w_1\rangle$ and $\langle \mathcal{M}_2, w_2\rangle$ be two pointed models. Then $\langle \mathcal{M}_1, w_1\rangle \equiv^{EF}_{\mathcal{ML}^m(\langle r\rangle)} \langle \mathcal{M}_2, w_2\rangle$ if and only if $\langle \mathcal{M}_1, w_1\rangle \leftrightarrows_{\mathcal{ML}^m(\langle r\rangle)} \langle \mathcal{M}_2, w_2\rangle$.*

Proof. For the right to left direction. Assume that $\langle S_1, w_1\rangle \sim \langle S_2, w_2\rangle$ is a bisimulation. We will prove that there is a strategy for Duplicator in the game $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$. First note that the game $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ is well defined, since by *(agree)* w_1 and w_2 are agreeing points. We show that there is a strategy for Duplicator by proving that (1) for any pair of tuples $\langle S, w\rangle$ and $\langle Q, v\rangle$ such that $\langle S, w\rangle \sim \langle Q, v\rangle$, and for any Spoiler step in the game $EF(\mathcal{M}_1[S], \mathcal{M}_2[Q], w, v)$, there is always an appropriate answer for Duplicator such that the next step of the game is $EF(\mathcal{M}_1[S'], \mathcal{M}_2[Q'], w', v')$ and $\langle S', w'\rangle \sim \langle Q', v'\rangle$. Given the initial assumptions, the fact that Duplicator has a winning strategy on the game $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ easily follows from (1). So let's suppose that $\langle S, w\rangle \sim \langle Q, v\rangle$ and consider the game $EF(\mathcal{M}_1[S], \mathcal{M}_2[Q], w, v)$. Without loss of generality, we assume that Spoiler chooses $\mathcal{M}_1$ to make his move. There are two kinds of moves Spoiler can do:

- Spoiler chooses to make a memorizing step, so the game goes on with $EF(\mathcal{M}_1[S \cup \{w\}], \mathcal{M}_2[Q \cup \{v\}], w, v)$. By the (*remember*) condition, we know that $\langle S \cup \{w\}, w\rangle \sim \langle Q \cup \{v\}, v\rangle$.

- Spoiler chooses to make a move step, so he chooses an R-successor w' of w. By the *(zig)* condition (we should use *(zag)* here if we would have supposed that Duplicator chooses $\mathcal{M}_2$ to make his move), there is an R-successor v' of v such that $\langle S, w'\rangle \sim \langle Q, v'\rangle$. Using *(agree)*, we know that w' and v' agree, so v' is a good choice for Duplicator, the next step of the game is $EF(\mathcal{M}_1[S], \mathcal{M}_2[Q], w', v')$ and $\langle S, w'\rangle \sim \langle Q, v'\rangle$.

For the other direction, let's suppose that Duplicator has a winning strategy $\mathcal{S}$ on the game $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$, where $\mathcal{M}_1 = \langle W_1, R_1, S_1, V_1\rangle$ and $\mathcal{M}_2 = \langle W_2, R_2, S_2, V_2\rangle$. We are going to define a bisimulation relation $\sim$ in the following way: $\langle S, w\rangle \sim \langle Q, v\rangle$ iff $EF(\mathcal{M}_1[S], \mathcal{M}_2[Q], w, v)$ is some stage of the game $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ reached by the players when Duplicator follows the strategy $\mathcal{S}$.

So now we have to see that the relation $\sim$ we defined is actually a bisimulation. Suppose that $\langle S, w\rangle \sim \langle Q, v\rangle$. Using the definition of $\sim$, that means that $EF(\mathcal{M}_1[S], \mathcal{M}_2[Q],\ w, v)$ is a reachable game state from $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ when Duplicator uses the strategy $\mathcal{S}$. Let's check the bisimulation conditions:

- The condition *(agree)* is easy to see, given that the definition of $\sim$ implies that if $\langle S, w\rangle \sim \langle Q, v\rangle$ then w and v are agreeing points.

- To see that the *(zig)* condition holds, suppose that $R^{\mathcal{M}}(m, m')$. One possible move for Spoiler in the game $EF(\mathcal{M}_1[S], \mathcal{M}_2[Q], w, v)$ is to choose m' from $\mathcal{M}_1$, and because Duplicator uses the winning strategy $\mathcal{S}$, he can answer with a point $v' \in \mathcal{M}_2$, a successor of v, such that w' and v' agree. Therefore, the next step of the game is $EF(\mathcal{M}_1[S], \mathcal{M}_2[Q], w',\ v')$, and by definition, $\langle S, w'\rangle \sim \langle Q, v'\rangle$. The *(zag)* condition is equivalent.

- For the $(remember)$ condition, note that in the game $EF(\mathcal{M}_1[S],\ \mathcal{M}_2[Q],\ w, v)$ Spoiler can choose to make a memorizing step, and therefore the next step of the game is $EF(\mathcal{M}_1[S \cup \{w\}], \mathcal{M}_2[Q \cup \{v\}], w, v)$. By definition, that means that $\langle S \cup \{w\}, w\rangle \sim \langle Q \cup \{v\}, v\rangle$.

Therefore, $\sim$ is actually a bisimulation. Because the game $EF(\mathcal{M}_1, \mathcal{M}_2, w_1,\ w_2)$ is a (trivial) reachable stage of itself, $\langle S_1, w\rangle \sim \langle S_2, v\rangle$ as desired. $\square$

The previous theorem shows that both notions are interchangeable, so using $\leftrightarrows_{\mathcal{L}}$ or $\equiv^{EF}_{\mathcal{L}}$ is going to be exactly the same from now on. But what really matters is that these notions *preserve* the truth values of formulas. Again, we are only going to give this proof for $\mathcal{ML}^m(\langle r\rangle)$, but the same result can be easily extended for the other memory fragments.

3.1.4. Theorem. *Let $\langle \mathcal{M}_1, w_1\rangle$ and $\langle \mathcal{M}_2, w_2\rangle$ be two pointed models. Then $\langle \mathcal{M}_1, w_1\rangle \equiv^{EF}_{\mathcal{ML}^m(\langle r\rangle)} \langle \mathcal{M}_2, w_2\rangle$ implies $\langle \mathcal{M}_1, w_1\rangle \leftrightsquigarrow_{\mathcal{ML}^m(\langle r\rangle)} \langle \mathcal{M}_2, w_2\rangle$.*

Proof. We prove that if w_1 and w_2 agree and Duplicator has a winning strategy on $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ then $\forall \varphi \in \mathcal{ML}^m(\langle r\rangle)$, $\mathcal{M}_1, w_1 \models \varphi \ \leftrightarrow\ \mathcal{M}_2, w_2 \models \varphi$. We proceed by induction on φ.

- The propositional and boolean cases are trivial.

- $\varphi = \textcircled{k}$. This case follows from the semantic definition in 2.1.3 and because w_1 and w_2 agree.

- $\varphi = \langle r \rangle \psi$. This is the standard modal case. Preservation is ensured thanks to the *move steps* in the definition of the game.

- $\varphi = \textcircled{r}\psi$. We prove that $\mathcal{M}_1, w_1 \models \textcircled{r}\psi$ implies $\mathcal{M}_2, w_2 \models \textcircled{r}\psi$. Suppose $\mathcal{M}_1, w_1 \models \textcircled{r}\psi$ then $\mathcal{M}_1[w_1], w_1 \models \psi$. The following claim is clear.

  Claim. Let $\mathcal{M}_1, \mathcal{M}_2$ be two models, $w_1 \in \mathcal{M}_1, w_2 \in \mathcal{M}_2$. If Duplicator has a winning strategy on $EF(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ then he has a winning strategy on $EF(\mathcal{M}_1[w_1], \mathcal{M}_2[w_2], w_1, w_2)$.

  Following this claim, Duplicator has a winning strategy on $EF(\mathcal{M}_1[w_1]$, $\mathcal{M}_2[w_2]$, w_1, $w_2)$. Applying inductive hypothesis and the fact that $\mathcal{M}_1[w_1]$, $w_1 \models \psi$, we conclude $\mathcal{M}_2[w_2], w_2 \models \psi$ and then $\mathcal{M}_2, w_2 \models \textcircled{r}\psi$. The other direction is identical.

This concludes the proof. □

One final remark. We have proved that bisimulations preserve satisfiability, but a valid question one can pose is: are these definitions too strong? That is, could we present weaker definitions that also preserve satisfiability? There is usually a mismatch between structural preserving morphisms and logic equivalence. This issue is not exclusive to the modal logic area, and it is the same mismatch one can find between partial isomorphisms and first order logic equivalence. For more information about this in the context of modal logic, see [BdRV01].

But there is something we can say in favor of the definitions we gave. In the class of image-finite models, we can establish the equivalence between bisimulations and logic equivalence. This is usually seen as a good sign that bisimulations are well defined.

3.1.5. Definition. A model $\mathcal{M}$ is *image finite* if for all points $w \in \mathcal{M}$, the set $\{v \mid wR^{\cup}v, \text{ where } R^{\cup} = \bigcup_{r \in \mathrm{REL}} R_r\}$ is finite.

We prove this property for the case of $\mathcal{ML}^m(\langle r \rangle)$, assuming a unique modal relation R_r. The cases for the other memory operators and the generalization for the multi-modal case are quite easy to establish.

3.1.6. Theorem. *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are image finite then $\langle \mathcal{M}_1, w_1 \rangle \leftrightsquigarrow_{\mathcal{ML}^m(\langle r \rangle)} \langle \mathcal{M}_2, w_2 \rangle$ implies $\langle \mathcal{M}_1, w_1 \rangle \equiv^{EF}_{\mathcal{ML}^m(\langle r \rangle)} \langle \mathcal{M}_2, w_2 \rangle$.*

Proof. We prove that the relation $\leftrightsquigarrow$ of modal equivalence induces, at any stage of the game, the next step in Duplicator's strategy (throughout this proof, we are going to work with $\mathcal{ML}^m(\langle r \rangle)$, so we will drop the subscripts in $\leftrightsquigarrow$ and $\equiv^{EF}$). We show that for any $\mathcal{M}'_1, \mathcal{M}'_2, w'_1 \in \mathcal{M}'_1, w'_2 \in \mathcal{M}'_2$ such that $\mathcal{M}'_1, w'_1 \leftrightsquigarrow \mathcal{M}'_2, w'_2$, and for any $v_1 \in \mathcal{M}'_1$ that Spoiler selects, Duplicator can always answer $v_2 \in \mathcal{M}'_2$ such that $\mathcal{M}'_1, v_1 \leftrightsquigarrow \mathcal{M}'_2, v_2$ (and symmetrically in the case Spoiler chooses $\mathcal{M}'_2$). Suppose Spoiler chooses $\mathcal{M}'_1$ to play.

1) Suppose Spoiler decides to take a *move step*. He chooses v_1, an R_r-successor of w'_1. Assume by contradiction that Duplicator cannot respond back. This means that he cannot exhibit v_2, an R_r-successor of w'_2, such that $\mathcal{M}'_1, v_1 \leftrightsquigarrow \mathcal{M}'_2, v_2$. Suppose $\{u_1, \ldots, u_n\}$ are the successors of w'_2. Let $\psi_1, \ldots, \psi_n$ be formulas such that $\mathcal{M}'_2, u_i \not\models \psi_i$ and $\mathcal{M}'_1, v_1 \models \psi_i$. It follows that $\mathcal{M}'_1, w'_1 \models \langle r\rangle(\psi_1 \wedge \cdots \wedge \psi_n)$ and $\mathcal{M}'_2, w'_2 \not\models \langle r\rangle(\psi_1 \wedge \cdots \wedge \psi_n)$. This is a contradiction, since we supposed $\mathcal{M}'_1, w'_1 \leftrightsquigarrow \mathcal{M}'_2, w'_2$.

2) Suppose Spoiler decides to remember w'_1. As in the above case, there are formulas $\psi_1, \ldots, \psi_n$ such that $\mathcal{M}'_1[w'_1], w'_1 \models \langle r\rangle(\psi_1 \wedge \cdots \wedge \psi_n)$ and $\mathcal{M}'_2[w'_2], w'_2 \not\models \langle r\rangle(\psi_1 \wedge \cdots \wedge \psi_n)$. By definition, $\mathcal{M}'_1, w'_1 \models \textcircled{\mathtt{r}}\langle r\rangle(\psi_1 \wedge \cdots \wedge \psi_n)$ and $\mathcal{M}'_2, w'_2 \not\models \textcircled{\mathtt{r}}\langle r\rangle(\psi_1 \wedge \cdots \wedge \psi_n)$ and we arrive to the same contradiction as in the previous case.

This concludes the proof. □

3.2 Expressive power

In this section we are going to compare the expressive power of different memory logic fragments. We want to make a comparison among them, and also with respect to both the basic modal logic and the hybrid logic $\mathcal{HL}(\downarrow)$, in order to have a general picture of where these fragments are situated. To do this, we will have to find a natural mapping between models of each logic, similar to the natural mapping that exists between Kripke models and first order models [BdRV01]. Such a mapping is easy to define in the case of the $\mathcal{ML}_\emptyset$ logics, where we only consider models with $S = \emptyset$: each Kripke model $\langle W, (R_r)_{r\in\text{REL}}, V\rangle$ can be identified with the memory model $\langle W, (R_r)_{r\in\text{REL}}, V, \emptyset\rangle$. Similarly, for sentences, the memory model $\langle W, (R_r)_{r\in\text{REL}}, V, \emptyset\rangle$ can be identified with the hybrid model $\langle W, (R_r)_{r\in\text{REL}}, V,\ g\rangle$ (for g arbitrary).

To improve the presentation of this section, sometimes we are going to present theorems that are later subsumed by stronger results. The reasons for doing this are in some cases just for the sake of clarity. In others it is because we believe that the proofs of some results are interesting by themselves.

We start by defining a way to compare the expressive power of two logics:

3.2.1. Definition. $[\mathcal{L} \leq \mathcal{L}']$ We say that $\mathcal{L}'$ is *at least as expressive as* $\mathcal{L}$ (notation $\mathcal{L} \leq \mathcal{L}'$) if there is a function Tr between formulas of $\mathcal{L}$ and $\mathcal{L}'$ such that for every model $\mathcal{M}$ and every formula φ of $\mathcal{L}$ we have that

$$\mathcal{M} \models_{\mathcal{L}} \varphi \text{ iff } \mathcal{M} \models_{\mathcal{L}'} \mathsf{Tr}(\varphi).$$

(here it should be understood that the model $\mathcal{M}$ is seen as a model of $\mathcal{L}$ on the left and as a model of $\mathcal{L}'$ on the right).

We say that $\mathcal{L}'$ is *strictly more expressive* than $\mathcal{L}$ (notation $\mathcal{L} < \mathcal{L}'$) if $\mathcal{L} \leq \mathcal{L}'$ but not $\mathcal{L}' \leq \mathcal{L}$. And we say that $\mathcal{L}$ and $\mathcal{L}'$ are equally expressive (notation $\mathcal{L} = \mathcal{L}'$) if $\mathcal{L} \leq \mathcal{L}'$ and $\mathcal{L}' \leq \mathcal{L}$.

3.2.1 Logics with an initially empty memory

Here we will compare the logics $\mathcal{ML}^m_\emptyset(\langle r\rangle)$ and $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ between them, and with the basic modal logic and the hybrid logic $\mathcal{HL}(\downarrow)$. Summarizing the results, we are going to establish that $\mathcal{ML} < \mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle) < \mathcal{ML}^m_\emptyset(\langle r\rangle) < \mathcal{HL}(\downarrow)$.

First we are going to show that the freedom to decide when to remember a point gives $\mathcal{ML}^m_\emptyset(\langle r\rangle)$ more expressive power when compared to $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$.

3.2.2. Theorem. $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle) < \mathcal{ML}^m_\emptyset(\langle r\rangle)$.

Proof. [$\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle) \leq \mathcal{ML}^m_\emptyset(\langle r\rangle)$]: It is easy to see that there is a translation Tr from $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ to $\mathcal{ML}^m_\emptyset(\langle r\rangle)$-formulas which maps $\langle\langle r\rangle\rangle\varphi$ to $\textcircled{r}\langle r\rangle\varphi$ and verifies $\mathcal{M} \models \varphi$ iff $\mathcal{M} \models \mathsf{Tr}(\varphi)$.

[$\mathcal{ML}^m_\emptyset(\langle r\rangle) \not\leq \mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$]: Let $\mathcal{M}_1 = \langle\{w, v, r\}, R_1, \emptyset, \emptyset\rangle$ and $\mathcal{M}_2 = \langle\{w, v, r\}, R_2, \emptyset, \emptyset\rangle$ such that $R_1 = \{(w, v), (v, r), (r, w)\}$, $R_2 = \{(w, v), (v, r), (r, v)\}$, as shown below:

$\mathcal{M}_1$ $\mathcal{M}_2$

We claim $\langle\mathcal{M}_1, w\rangle \equiv^{EF}_{\mathcal{ML}^m(\langle\langle r\rangle\rangle)} \langle\mathcal{M}_2, w\rangle$. As every point in both models has a unique successor, Duplicator has only one way of playing, which is actually a winning strategy. Hence $\langle\mathcal{M}_1, w\rangle \equiv_{\mathcal{ML}^m(\langle\langle r\rangle\rangle)} \langle\mathcal{M}_2, w\rangle$. But $\mathcal{M}_1, w \not\models \langle r\rangle\textcircled{r}\langle r\rangle\langle r\rangle\textcircled{k}$, while $\mathcal{M}_2, w \models \langle r\rangle\textcircled{r}\langle r\rangle\langle r\rangle\textcircled{k}$. □

We will now compare the expressive power of memory logics with the basic modal logic $\mathcal{ML}$. It is not difficult to see intuitively that $\textcircled{r}$ and $\textcircled{k}$ do bring additional expressive power into the language of $\mathcal{ML}$: with their help we can detect cycles in a given model, while formulas of $\mathcal{K}$ are invariant under unraveling.

3.2.3. Theorem. $\mathcal{ML} < \mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$.

Proof. It is quite easy to see that $\mathcal{ML} \leq \mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ taking Tr to be a translation that maps $\langle r\rangle$ to $\langle\langle r\rangle\rangle$, and that acts as the identity function for the rest of the operators.

To see that $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle) \not\leq \mathcal{ML}$, let $\mathcal{M}_1 = \langle\{w\}, \{(w, w)\}, \emptyset\rangle$ and $\mathcal{M}_2 = \langle\{u, v\}, \{(u, v), (v, u)\}, \emptyset\rangle$ be two Kripke models as shown below:

$\mathcal{M}_1$ $\mathcal{M}_2$

It is easy to see that both models are $\mathcal{ML}$ bisimilar using the bisimulation with the pairs $(w,u),(w,v)$. However, they can be distinguished by the $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formula $\langle\langle r\rangle\rangle$ⓚ. □

We will now compare the expressive power of memory logics with respect to hybrid logics. The most natural choice for the comparison is the hybrid logic $\mathcal{HL}(\downarrow)$. We will prove that $\mathcal{HL}(\downarrow)$ is strictly more expressive than $\mathcal{ML}^m_\emptyset(\langle r\rangle)$. Intuitively, $\downarrow$ can easily simulate ⓡ, but ⓚ does not distinguish between different memorized points (while nominals bound by $\downarrow$ do).

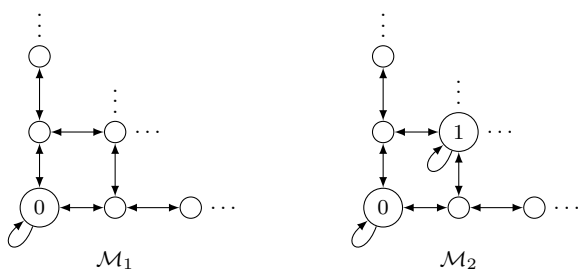
3.2.4. Theorem. $\mathcal{ML}^m_\emptyset(\langle r\rangle) < \mathcal{HL}(\downarrow)$.

Proof. We first prove that $\mathcal{ML}^m_\emptyset(\langle r\rangle) \leq \mathcal{HL}(\downarrow)$. We define the translation Tr, taking $\mathcal{ML}^m_\emptyset(\langle r\rangle)$-formulas over the signature $\langle$PROP, REL$\rangle$ to $\mathcal{HL}(\downarrow)$ sentences over the signature $\langle$PROP, REL, NOM$\rangle$. Tr is defined for any finite set $N \subseteq$ NOM as follows:

$$\begin{array}{rcl}
\mathsf{Tr}_N(p) & = & p \quad p \in \text{PROP} \\
\mathsf{Tr}_N(ⓚ) & = & \bigvee_{i\in N} i \\
\mathsf{Tr}_N(\neg\varphi) & = & \neg\mathsf{Tr}_N(\varphi) \\
\mathsf{Tr}_N(\varphi_1 \wedge \varphi_2) & = & \mathsf{Tr}_N(\varphi_1) \wedge \mathsf{Tr}_N(\varphi_2) \\
\mathsf{Tr}_N(\langle r\rangle\varphi) & = & \langle r\rangle\mathsf{Tr}_N(\varphi) \\
\mathsf{Tr}_N(ⓡ\varphi) & = & \downarrow i.\mathsf{Tr}_{N\cup\{i\}}(\varphi) \quad \text{where } i \notin N.
\end{array}$$

A simple induction shows that $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}, g, w \models \mathsf{Tr}_\emptyset(\varphi)$, for any g.

Now we prove that $\mathcal{HL}(\downarrow)$ is strictly more expressive than $\mathcal{ML}^m_\emptyset(\langle r\rangle)$. Let $\mathcal{M}_1 = \langle\omega, R_1, \emptyset, \emptyset\rangle$ and $\mathcal{M}_2 = \langle\omega, R_2, \emptyset, \emptyset\rangle$, where $R_1 = \{(n,m) \mid n \neq m\} \cup \{(0,0)\}$ and $R_2 = R_1 \cup \{(1,1)\}$. Graphically,



$\mathcal{M}_1$ $\mathcal{M}_2$

where the accessibility relation is the transitive closure of the arrows shown but without reflexive loops excepts those explicitly marked.

We prove that $\langle \mathcal{M}_1, 0\rangle \equiv^{EF}_{\mathcal{ML}^m_\emptyset(\langle r\rangle)} \langle \mathcal{M}_2, 0\rangle$ showing a winning strategy for Duplicator. Intuitively, the strategy for Duplicator is as follows: whenever one player is in $\langle \mathcal{M}_1, 0\rangle$ the other will be in $\langle \mathcal{M}_2, 0\rangle$ or $\langle \mathcal{M}_2, 1\rangle$, and conversely whenever a player is in $\langle \mathcal{M}_1, n\rangle$, $n > 0$, the other will be in $\langle \mathcal{M}_2, m\rangle$, $m > 1$. This is maintained until Spoiler (if ever) decides to remember a point. Once this is done, then any move leads to a win of Duplicator.

Formally, the winning strategy will have two stages:

1. While Spoiler does not remember any reflexive point, Duplicator plays as follows: if Spoiler chooses 0 in any model, Duplicator chooses 0 in the other; if Spoiler chooses $n > 0$ in $\mathcal{M}_1$, Duplicator plays $n + 1$ in $\mathcal{M}_2$; if Spoiler chooses $n > 0$ in $\mathcal{M}_2$, Duplicator plays $n - 1$ in $\mathcal{M}_1$. Notice that with this strategy Spoiler chooses a reflexive point if and only if Duplicator answers with a reflexive one. This is clearly a winning strategy.

2. If ever Spoiler decides to remember a reflexive point, Duplicator starts using the following strategy: if Spoiler selects a point n, Duplicator answers with an agreeing point m of the opposite model. Notice that this is always possible since both n and m see infinitely many non-memorized points and at least one memorized point.

On the other hand, let φ be the formula $\downarrow i.\langle r\rangle(i \wedge \langle r\rangle(\neg i \wedge \downarrow i.\langle r\rangle i))$. It is easy to see that $\mathcal{M}_1, 0 \not\models \varphi$ but $\mathcal{M}_2, 0 \models \varphi$. □

We have shown that $\mathcal{ML}^m_\emptyset(\langle r\rangle) < \mathcal{HL}(\downarrow)$ but the proof seems to intrinsically use infinite models, in contrast with the proofs for Theorems 3.2.2, 3.2.3 and 3.2.4 in which finite models are used. Actually, $\mathcal{ML}^m_\emptyset(\langle r\rangle) < \mathcal{HL}(\downarrow)$ even on *finite models*. For this purpose we will first introduce a version of the Ehrenfeucht-Fraïssé game presented in Definition 3.1.2 where the number of turns is bounded.

3.2.5. Definition. The *n-moves Ehrenfeucht-Fraïssé game* for a given logic $\mathcal{L}$, denoted $EF^n(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$, is the game in which Spoiler can only make n moving steps in the game to beat Duplicator. In the case Duplicator has a strategy to remain undefeated for n steps, he wins the game and we will write $\langle \mathcal{M}_1, w_1\rangle \equiv^{EF^n} \langle \mathcal{M}_2, w_2\rangle$.

We use the usual definition of modal depth, that essentially counts the maximum nesting of modalities. Anyway, we define it here for the sake of completeness for the whole set of operators we presented:

3.2.6. Definition. We define the *modal depth* of modal formulas as follows:

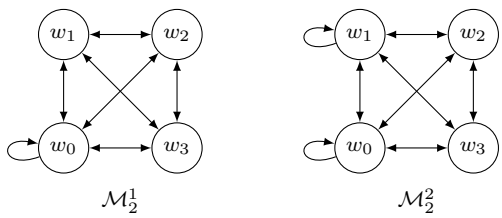


Figure 3.1: Spoiler cannot win with only 2 moves.

$$
\begin{array}{rcl}
depth(p) & = & 0 \\
depth(\textcircled{k}) & = & 0 \\
depth(\neg\varphi) & = & depth(\varphi) \\
depth(\varphi \wedge \psi) & = & max\{depth(\varphi), depth(\psi)\} \\
depth(\langle r\rangle\varphi) & = & 1 + depth(\varphi) \\
depth(\langle\langle r\rangle\rangle\varphi) & = & 1 + depth(\varphi) \\
depth(\textcircled{r}\varphi) & = & depth(\varphi) \\
depth(\textcircled{e}\varphi) & = & depth(\varphi) \\
depth(\textcircled{f}\varphi) & = & depth(\varphi)
\end{array}
$$

There is an obvious connection between rounds in a game and modal depth of a formula. We will state without a proof the following easy theorem.

3.2.7. Theorem. *For any pair of pointed models, $\langle \mathcal{M}_1, w_1\rangle \equiv^{EF^n} \langle \mathcal{M}_2, w_2\rangle$ if and only if for every formula φ of $\mathcal{L}_i$ with modal depth n, $\mathcal{M}_1, w_1 \models \varphi$ iff $\mathcal{M}_2, w_2 \models \varphi$.*

Now we can prove the desired result for finite models:

3.2.8. Theorem. *$\mathcal{ML}^m_\emptyset(\langle r\rangle) < \mathcal{HL}(\downarrow)$ over the class of finite models.*

Proof. We will prove that there is a property φ expressible in $\mathcal{HL}(\downarrow)$ that cannot be expressed in $\mathcal{ML}^m_\emptyset(\langle r\rangle)$ over finite models. To do this, for every n we will exhibit two finite models $\mathcal{M}^1_n, \mathcal{M}^2_n$ such that $\mathcal{M}^1_n, w_0 \models \varphi$, $\mathcal{M}^2_n, w_0 \not\models \varphi$ but $\langle \mathcal{M}^1_n, w_0\rangle \equiv^{EF^n}_{\mathcal{ML}^m_\emptyset(\langle r\rangle)} \langle \mathcal{M}^2_n, w_0\rangle$. This implies that there is no $\mathcal{ML}^m_\emptyset(\langle r\rangle)$-formula ψ capable of expressing this property.

Let $\varphi = \downarrow i.\langle r\rangle(i \wedge \langle r\rangle(\neg i \wedge \downarrow i.\langle r\rangle i))$ as in the proof of Theorem 3.2.4, and let, for $n \geq 1$, $\mathcal{M}^1_n = \langle W_n, R_1, \emptyset, \emptyset\rangle$ and $\mathcal{M}^2_n = \langle W_n, R_2, \emptyset, \emptyset\rangle$ where $W_n = \{w_0, \ldots, w_{n+1}\}$, $R_1 = \{(n, m) \mid n \neq m\} \cup \{(w_0, w_0)\}$, and $R_2 = R_1 \cup \{(w_1, w_1)\}$ ($\mathcal{M}^1_2$ and $\mathcal{M}^2_2$ are shown in Figure 3.1). Clearly, for every $n \geq 1$, $\mathcal{M}^1_n, w_0 \not\models \varphi$ and $\mathcal{M}^2_n, w_0 \models \varphi$.

To prove that $\langle \mathcal{M}^1_n, w_0\rangle \equiv^{EF}_n \langle \mathcal{M}^2_n, w_0\rangle$, we will describe Duplicator's winning strategy:

1. While Spoiler does not remember any reflexive point, Duplicator plays with the following strategy: whenever Spoiler is in w_k, $2 \leq k \leq n{+}1$ in one model, Duplicator is in an agreeing point $w_{k'}$, $2 \leq k' \leq n+1$ in the other one. If one player is in w_0 in $\mathcal{M}_n^1$ then the other is in w_0 or w_1 in $\mathcal{M}_n^2$. Finally, if Spoiler plays w_1 in $\mathcal{M}_n^1$, Duplicator plays in an agreeing w_k, $2 \leq k \leq n+1$ in $\mathcal{M}_n^2$. Note that with this strategy, Spoiler chooses a reflexive point iff Duplicator answers with a reflexive one, and that Duplicator will always be able to choose an agreeing point.

2. If ever Spoiler decides to remember a reflexive point, then for every point w_i chosen by Spoiler, Duplicator will always have an agreeing point w_j on the opposite model he can choose from. This happens because the models have $n+2$ points, and therefore there is always at least two non-memorized points. At each round the number of non-memorized points can only be decremented by one, and then up to round n both players will always see memorized and not memorized points from w_i and well as from w_j.

□

The $\mathcal{HL}(\downarrow)$-sentence we use in the proofs of Theorem 3.2.4 and 3.2.8 has only one nominal. Hence, we have actually proved that $\mathcal{HL}_1(\downarrow) \not\leq \mathcal{ML}_\emptyset^m(\langle r\rangle)$, where $\mathcal{HL}_1(\downarrow)$ is $\mathcal{HL}(\downarrow)$ restricted to only one nominal. But actually, it is also the case that $\mathcal{ML}_\emptyset^m(\langle r\rangle) \not\leq \mathcal{HL}_1(\downarrow)$. More generally, for any fixed number k of nominals, the logics $\mathcal{HL}_k(\downarrow)$ and $\mathcal{ML}_\emptyset^m(\langle r\rangle)$ are incomparable.

3.2.9. Theorem. *For any fixed k, the logics $\mathcal{HL}_k(\downarrow)$ and $\mathcal{ML}_\emptyset^m(\langle r\rangle)$ are incomparable in terms of expressive power.*

Proof. We will show the proof for $k = 1$, the general case being similar. $\mathcal{HL}_1(\downarrow) \not\leq \mathcal{ML}_\emptyset^m(\langle r\rangle)$ is a direct consequence of the proof of Theorem 3.2.4.

To prove $\mathcal{ML}_\emptyset^m(\langle r\rangle) \not\leq \mathcal{HL}_1(\downarrow)$, let $\mathcal{M}_1 = \langle\{1,2,3\}, \{(i,j) \mid 1 \leq i,j \leq 3\}, \emptyset, \emptyset\rangle$ and $\mathcal{M}_2 = \langle\{1,2\}, \{(i,j) \mid 1 \leq i,j \leq 2\}, \emptyset, \emptyset\rangle$ (a clique of size 3 and 2 respectively). It is easy to check that $\langle\mathcal{M}_1, 1\rangle \leftrightarrows_{\mathcal{HL}_1(\downarrow)} \langle\mathcal{M}_2, 1\rangle$. However, the formula $\varphi = \textcircled{r}\langle r\rangle(\neg\textcircled{k} \wedge \textcircled{r}\langle r\rangle(\neg\textcircled{k} \wedge \textcircled{r}\langle r\rangle\neg\textcircled{k}))$ distinguishes the models: $\mathcal{M}_1, 1 \models \varphi$ but $\mathcal{M}_2, 1 \not\models \varphi$.

The proof for $\mathcal{HL}_k(\downarrow)$ is similar, taking cliques of the appropriate size. □

3.2.2 Erase and Forget operators

Now we want to include the operators ⓔ and ⓕ in the picture, and see which is the expressive power associated to them. We are only going to consider here ⓔ and ⓕ in the context of the class $\mathcal{C}_\emptyset$, and with the usual interpretation for the diamond operator.

The first result regarding these new operators says that independently adding ⓔ and ⓕ does increase the expressive power.

3.2.10. Theorem. *$\mathcal{ML}^m_\emptyset(\langle r\rangle) < \mathcal{ML}^m_\emptyset(\langle r\rangle, ⓕ)$ and $\mathcal{ML}^m_\emptyset(\langle r\rangle) < \mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ)$.*

Proof. It is trivial to see that $\mathcal{ML}^m_\emptyset(\langle r\rangle) \leq \mathcal{ML}^m_\emptyset(\langle r\rangle, ⓕ), \mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ)$ taking the identity translation. To see that $\mathcal{ML}^m_\emptyset(\langle r\rangle) \neq \mathcal{ML}^m_\emptyset(\langle r\rangle, ⓕ)$ and $\mathcal{ML}^m_\emptyset(\langle r\rangle) \neq \mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ)$, let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the models described in the proof of Theorem 3.2.4. Recall that $\langle \mathcal{M}_1, 0\rangle$ is $\mathcal{ML}^m_\emptyset(\langle r\rangle)$-bisimilar to $\langle \mathcal{M}_2, 0\rangle$.

- To see that $\mathcal{ML}^m_\emptyset(\langle r\rangle) \neq \mathcal{ML}^m_\emptyset(\langle r\rangle, ⓕ)$, we show that these two pointed models are distinguishable with an $\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓕ)$-formula. Let

  $$\psi = [r]ⓕ(\langle r\rangle(ⓚ \wedge \langle r\rangle ⓚ))$$

  saying "no matter which accessible point I choose, I can move to it, then forget it and finally move again to a memorized point connected with a memorized point". Now let

  $$\varphi = ⓡ\langle r\rangle(\neg ⓚ \wedge ⓡ\psi)$$

  It is clear that $\mathcal{M}_2, 0 \models \varphi$, since one can remember the point 0, then move to point 1 (which is not memorized), and remember it leaving the model in the point $\mathcal{M}_2[0,1]$ and the evaluating point in 1. Then it is easy to see that $\mathcal{M}_2[0,1], 1 \models \psi$. However, one can verify that $\mathcal{M}_1, 0 \not\models \varphi$. Indeed, suppose that, after remembering the point 0, we move to point $n > 0$ and we remember it. By the definition of $\mathcal{M}_1$, the point n will not be reflexive. Now, $\mathcal{M}_1[0,n], n \not\models \psi$ because $\mathcal{M}_1[0,n], 0 \not\models ⓕ(\langle r\rangle(ⓚ \wedge \langle r\rangle ⓚ))$, i.e. $\mathcal{M}_1[n], 0 \not\models \langle r\rangle(ⓚ \wedge \langle r\rangle ⓚ)$.

- Showing that $\mathcal{ML}^m_\emptyset(\langle r\rangle) \neq \mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ)$ is a bit more easy. Define the formulas $\psi' = ⓡ\langle r\rangle ⓚ$ and $\varphi' = ⓡ\langle r\rangle(\neg ⓚ \wedge ⓔ\psi')$. It is not difficult to see that $\mathcal{M}_2, 0 \models \varphi'$ but $\mathcal{M}_1, 0 \not\models \varphi'$.

□

On the other hand, we are still below the expressive power of $\mathcal{HL}(\downarrow)$:

3.2.11. Theorem. *$\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ, ⓕ) \leq \mathcal{HL}(\downarrow)$.*

Proof. In the line of the proof of Theorem 3.2.4, we define a truth-preserving translation from formulas of $\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ, ⓕ)$ into formulas of $\mathcal{HL}(\downarrow)$. To define our translation we use a finite sequence S of nominals in NOM, where each nominal i in the sequence is tagged with a superscript r (representing a remember) or with

a superscript f (representing a forget). We use the operation $S \circ i$ to denote the operation of inserting the element i at the end of the sequence S.

$$\begin{array}{rcl}
\mathsf{Tr}_S(p) & = & p \quad p \in \textsc{prop} \\
\mathsf{Tr}_S(\neg\varphi) & = & \neg\mathsf{Tr}_S(\varphi) \\
\mathsf{Tr}_S(\varphi_1 \wedge \varphi_2) & = & \mathsf{Tr}_S(\varphi_1) \wedge \mathsf{Tr}_S(\varphi_2) \\
\mathsf{Tr}_S(\langle r\rangle\varphi) & = & \langle r\rangle\mathsf{Tr}_S(\varphi) \\
\mathsf{Tr}_s(\textcircled{r}\varphi) & = & \downarrow i.\mathsf{Tr}_{S\circ\{i^r\}}(\varphi) \quad \text{where } i \notin S. \\
\mathsf{Tr}_s(\textcircled{f}\varphi) & = & \downarrow i.\mathsf{Tr}_{S\circ\{i^f\}}(\varphi) \quad \text{where } i \notin S. \\
\mathsf{Tr}_S(\textcircled{k}) & = & T(S)
\end{array}$$

Here, T is a translation from sequences of nominals to $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e}, \textcircled{f})$-formulas is defined in the following way:

$$\begin{array}{rcl}
T(\lambda) & = & \bot \\
T(S \circ i^r) & = & i \vee T(S) \\
T(S \circ i^f) & = & \neg i \wedge T(S)
\end{array}$$

A simple induction shows that $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}, g, w \models \mathsf{Tr}_\lambda(\varphi)$, for any g. □

3.2.12. Corollary. *$\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e}) \leq \mathcal{HL}(\downarrow)$ and $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{f}) \leq \mathcal{HL}(\downarrow)$.*

Now we will show that the logic $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e})$ is not more expressive than the logic $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{f})$. The way to show this is following a game argument, as we did for the case of $\mathcal{ML}^m_\emptyset(\langle r\rangle)$.

3.2.13. Theorem. *$\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{f}) \not\leq \mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e})$.*

Proof. Let $\mathcal{M} = \langle \{s\} \cup \omega_0 \cup \omega_1 \cup \ldots, R, \emptyset, \emptyset\rangle$, where each ω_i is a different copy of ω, and $R = \{(n,m) \mid n \in \omega_i, m \in \omega_j, i \leq j\} \cup \{(n,s),(s,n) \mid \text{for all } n \neq s\}$. We can imagine this model as the sequence of the natural numbers with the $\leq$ relationship between them and with a spy point s. But each natural number n is actually an infinite reflexive clique in which every point there is connected with every other point that belongs to a greater clique.

We prove that $\langle \mathcal{M}, w_0\rangle \equiv^{EF} \langle \mathcal{M}, w_1\rangle$ for $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e})$, where $w_0 \in \omega_0$ and $w_1 \in \omega_1$. Given a point w, we define the *neighborhood* of w as $N(w) = \{v \mid wRv\}$, and we say that the neighborhood of a point w *is memorized* when $N(w) \cap S \neq \emptyset$. The strategy we are going to define observes the following invariant:

- Every time Spoiler has moved to a point w, then Duplicator has answered with an agreeing point v such that $N(w)$ was not memorized if and only if $N(v)$ was not memorized.

- Every time Spoiler has moved to a point $w \in \omega_i$, Duplicator has answered with a point $v \in \omega_j$. And every time Spoilers has moved to s, Duplicator has moved to s.

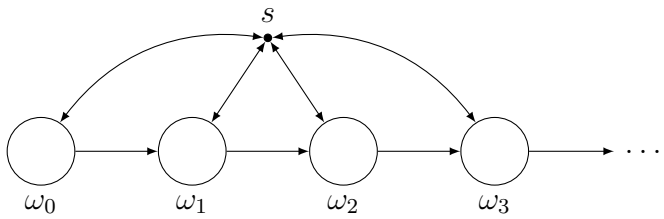


Figure 3.2: The model $\mathcal{M}$. The transitive connections between the cliques ω_i are not shown.

It is clear that this invariant holds at the beginning of the game. So let's see that each step of the strategy preserves the invariant. One general remark: at any stage of the game, the number of memorized points is always finite. Assume that Spoiler is in a point $w \in \omega_s$ and Duplicator in $v \in \omega_d$. The strategy for Duplicator is the following:

1. If Spoiler decides to remember w, then the game continues with both w and v memorized. So both $N(w)$ and $N(v)$ become memorized, and the invariant is preserved.

2. If Spoiler decides to forget all the points in the model, then both $N(w)$ and $N(v)$ become not memorized, and the game continues with the invariant preserved.

3. If Spoiler moves to s in one model, Duplicator moves to s in the other model. Since every point of every ω_i is connected to s, this is always a possible move for Duplicator. Given the invariant and the fact that s is connected with every other point in the model, it is easy to see that $N(s)$ is not memorized in one model iff $N(s)$ is not memorized in the other model.

4. If Spoiler plays in a point $w' \in \omega_{s'}$ such that $N(w')$ is not memorized, then Duplicator chooses a clique $\omega_{d'}$ and a point $v' \in \omega_{d'}$ such that $N(v')$ is not memorized. Note that by definition of neighborhood and the fact that every ω_i is a reflexive clique, w' and v' are not in S. Furthermore, this is always a valid move for Duplicator, given that there are infinitely many cliques ω_i connected with ω_d and the fact that the number of memorized points is finite. So Duplicator can always choose a sufficiently large d' and a point $v' \in \omega_{d'}$ such that $N(v')$ is not memorized.

5. If Spoiler plays in a point $w' \in \omega_{s'}$ such that $N(w')$ is memorized, then Spoiler moves to an agreeing point $v' \in \omega_{d'}$. Let's see that there is always such v' and that the invariant is preserved. If $N(w)$ is not memorized, given the shape of the model, the only possibility is that $w' = s$. Therefore $v' = s$,

and we have already seen that the neighborhoods match in this case. The other case is that $N(w)$ is memorized. Given the invariant, we know that $N(v)$ is memorized, so if Spoiler chooses $w' \in S$, we know that there is a $v' \in S$ that Spoiler can move to. In this case it is trivial to see that $N(v')$ is not memorized. On the other hand, if Spoiler chooses $w' \notin S$, then a safe choice for Duplicator is a non-memorized $v' \in \omega_d$, that is, a point in the same clique as v. Since each ω_i is infinite, there is always such a v', and also this choice guarantees that $N(v')$ is not memorized.

On the other hand let $\varphi = \textcircled{r}\langle r\rangle\langle r\rangle(\neg\textcircled{k} \wedge \langle r\rangle\textcircled{k} \wedge \textcircled{r}\langle r\rangle(\textcircled{k} \wedge \textcircled{f}[r]\neg\textcircled{k}))$ be a formula of $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{f})$. It is easy to see that $\mathcal{M}, w_1 \models \varphi$ but $\mathcal{M}, w_0 \not\models \varphi$. □

3.2.14. Corollary. *$\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e}) < \mathcal{HL}(\downarrow)$ and $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e}) < \mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e}, \textcircled{f})$*

Proof. Trivial given Theorems 3.2.13 and 3.2.11. □

To end this section, we want to observe that there are still some interesting questions that remain open. For example, the relation between $\mathcal{HL}(\downarrow)$ and $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e}, \textcircled{f})$:

3.2.15. Question. $\mathcal{HL}(\downarrow) \neq \mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e}, \textcircled{f})$?

We believe that the answer to this question is positive, but we have not found yet a pair of models (in the same direction as Theorems 3.2.13 and 3.2.4) in which the difference can be shown. The other natural question is the relation between $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e})$ and $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{f})$:

3.2.16. Question. $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e}) \not\leq \mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{f})$?

The proof for this should also follow the same style as Theorem 3.2.13.

3.2.3 Memory logics with a stack

Here we want to analyze other memory containers different than a set. It seems quite easy to see that replacing sets by lists gives us the same expressive power as $\mathcal{HL}(\downarrow)$. Another "classic" data structure is the stack, and in this case it is not clear *a priori* how the resulting logic will behave in terms of expressive power. We want to analyze then the outcome of using a stack as a storage structure. Note that if we change the type of memory container we should also change the memory operators accordingly. Therefore we will replace $\textcircled{r}$ and $\textcircled{k}$ by the usual stack operators. Formally, the syntax is:

$$\text{FORMS} ::= \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle r\rangle\varphi \mid \langle\langle r\rangle\rangle\varphi \mid (\texttt{push})\varphi \mid (\texttt{pop})\varphi \mid (\texttt{top})$$

where $p \in \text{PROP}$, $r \in \text{REL}$ and $\varphi, \varphi_1, \varphi_2 \in \text{FORMS}$. The semantics is the same for the already presented operators, plus the following rules for the new stack operators:

$$\begin{array}{rcl}
\mathcal{M}, w \models (\texttt{push})\varphi & \text{iff} & \mathcal{M}[w], w \models \varphi \\
\mathcal{M}, w \models (\texttt{pop})\varphi & \text{iff} & \langle W, (R_r)_{r\in\text{REL}}, V, S'\rangle, w \models \varphi \\
 & & \text{for some } S', w' \text{ such that } S = S' \cdot [w'] \\
\mathcal{M}, w \models (\texttt{top}) & \text{iff} & S = S' \cdot [w] \text{ for some } S'
\end{array}$$

Given a model $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ and a list of points $[w_1, \ldots, w_n]$, $w_i \in W$, we define $\mathcal{M}[w_1, \ldots, w_n] = \langle W, (R_r)_{r\in\text{REL}}, V, S \cdot [w_1, \ldots, w_n]\rangle$, where $\cdot$ is the concatenation operation on lists. We call this logic $\mathcal{ML}^{st}_{\emptyset}(\langle r\rangle)$.

Here we are going to focus in the logic that starts evaluating a formula in models with an empty stack and with the usual diamond operator. We will compare $\mathcal{ML}^{st}_{\emptyset}(\langle r\rangle)$ with $\mathcal{HL}(\downarrow)$ and prove that they are equivalently expressive. This might come as a surprise, as we could think that the restricted access to the elements in the stack might actually limit the expressive power of $\mathcal{ML}^{st}_{\emptyset}(\langle r\rangle)$. The proof below shows that it is the possibility to 'make copies of the stack' while evaluating a formula what solves the problem.

3.2.17. THEOREM. *$\mathcal{ML}^{st}_{\emptyset}(\langle r\rangle) = \mathcal{HL}(\downarrow)$.*

PROOF. To prove $\mathcal{ML}^{st}_{\emptyset}(\langle r\rangle) \leq \mathcal{HL}(\downarrow)$, we define the translation mapping an $\mathcal{ML}^{st}_{\emptyset}(\langle r\rangle)$-formula and a list of nominals N into an $\mathcal{HL}(\downarrow)$-formula.

$$\begin{array}{rcl}
\mathsf{Tr}_N(p) & = & p \quad p \in \text{PROP} \\
\mathsf{Tr}_N(\neg\varphi) & = & \neg\mathsf{Tr}_N(\varphi) \\
\mathsf{Tr}_N(\langle r\rangle\varphi) & = & \langle r\rangle\mathsf{Tr}_N(\varphi) \\
\mathsf{Tr}_N(\varphi_1 \wedge \varphi_2) & = & \mathsf{Tr}_N(\varphi_1) \wedge \mathsf{Tr}_N(\varphi_2) \\
\mathsf{Tr}_N((\texttt{push})\varphi) & = & \downarrow i.\mathsf{Tr}_{N\cdot i}(\varphi) \quad \text{where } i \notin N \\
\mathsf{Tr}_{N\cdot i}((\texttt{pop})\varphi) & = & \mathsf{Tr}_N(\varphi) \\
\mathsf{Tr}_{[\,]}((\texttt{pop})\varphi) & = & \neg\top \\
\mathsf{Tr}_{N\cdot i}((\texttt{top})) & = & i \\
\mathsf{Tr}_{[\,]}((\texttt{top})) & = & \neg\top
\end{array}$$

Let $[\,]$ be the empty list, we can show by induction on φ that $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}, g, w \models \mathsf{Tr}_{[\,]}(\varphi)$, for any g.

To prove $\mathcal{HL}(\downarrow) \leq \mathcal{ML}^{st}_{\emptyset}(\langle r\rangle)$ we define a translation mapping an $\mathcal{HL}(\downarrow)$-formula and a list of nominals N into an $\mathcal{ML}^{st}_{\emptyset}(\langle r\rangle)$-formula. The translation coincides with the translation above for the propositional, negation, conjunction and modality cases. We translate $\downarrow$ and nominals as follows:

$$\begin{array}{rcl}
\mathsf{Tr}_N(\downarrow i.\varphi) & = & (\texttt{push})\mathsf{Tr}_{N\cdot i}(\varphi) \\
\mathsf{Tr}_N(i) & = & (\texttt{pop})^{|N|-n}(\texttt{top}) \quad i \in \text{NOM}, N[n] = i, \forall m > n : N[m] \neq i,
\end{array}$$

where $|N|$ represents the length of N and $N[n]$ represents the n-th element of N. It can be shown by induction in φ that if φ is an $\mathcal{HL}(\downarrow)$-sentence, $\mathcal{M}, g, w \models \varphi$ iff $\mathcal{M}, w \models \mathsf{Tr}_{[\ ]}(\varphi)$ for any g. $\square$

One natural question turns up here. Are there minimal conditions one can impose to the storage structure (and its operators) in order to have a memory logic equivalent to $\mathcal{HL}(\downarrow)$? It is clear that having a complete mapping of memorized points is enough (as it is the case if we use a list). The case for the stack was not that clear initially, but the ability to "clone" the current stack gave us the capacity to keep a complete mapping of points. Establishing general conditions does not seems easy, since the container and its associated operators may have, *a priori*, any kind of structure and behavior. Being able to store and retrieve memorized elements in an ordered way seems to be a necessary condition, but it is not trivial to formalize this notion in a general way. We are not going to go into more details here, but further research can follow this direction.

3.2.4 Comparing apples with oranges

Comparing the expressive power between the logics that start evaluating formulas in $\mathcal{C}_\emptyset$ and the ones that use an arbitrary memory poses a complication because, strictly speaking, each of them uses a different class of models. It is not completely clear whether it is fair to make the comparison at all, but once one is convinced to make it, the next step is to decide how to define the mapping between each type of models. The most natural option seems to involve a shift in the signature of the language, in order to preserve the information stored in the models.

We start by discussing the relation among the memory logics that use a set as the storage structure.

3.2.18. Theorem.

1. *The logic $\mathcal{ML}^m_\emptyset(\langle r\rangle)$ over the signature $\langle \text{PROP} \cup \{\mathit{known}\}, \text{REL}\rangle$ is equivalent to the logic $\mathcal{ML}^m(\langle r\rangle)$ over the signature $\langle \text{PROP}, \text{REL}\rangle$*

2. *The logic $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ over the signature $\langle \text{PROP} \cup \{\mathit{known}\}, \text{REL}\rangle$ is equivalent to the logic $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ over the signature $\langle \text{PROP}, \text{REL}\rangle$*

Proof. The argument for item 2 is exactly the same as the one for item 1. Hence, let us prove $\mathcal{ML}^m_\emptyset(\langle r\rangle) = \mathcal{ML}^m(\langle r\rangle)$ (over the appropriate signatures).

We start by associating every model $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ of the logic $\mathcal{ML}^m(\langle r\rangle)$ over $\langle \text{PROP}, \text{REL}\rangle$ with the model $\mathcal{M}' = \langle W, (R_r)_{r\in\text{REL}}, V', \emptyset\rangle$ of the logic $\mathcal{ML}^m_\emptyset(\langle r\rangle)$ over $\langle \text{PROP}\cup\{\mathit{known}\}, \text{REL}\rangle$ where V' is identical to V over PROP and $V'(\mathit{known}) = S$.

$[\mathcal{ML}^m_\emptyset(\langle r\rangle) \leq \mathcal{ML}^m(\langle r\rangle)]$: use the translation Tr that only rewrites the propositional symbol *known* as ⓚ in any formula of $\mathcal{ML}^m_\emptyset(\langle r\rangle)$. Clearly for any formula $\varphi \in \mathcal{ML}^m_\emptyset(\langle r\rangle)$ we have that $\mathcal{M}', w \models \varphi$ iff $\mathcal{M}, w \models \mathsf{Tr}(\varphi)$.

$[\mathcal{ML}^m(\langle r\rangle) \leq \mathcal{ML}^m_\emptyset(\langle r\rangle)]$: use the translation Tr that only rewrites ⓚ as (ⓚ $\vee$ *known*) in any formula of $\mathcal{ML}^m(\langle r\rangle)$. Clearly for any formula $\varphi \in \mathcal{ML}^m(\langle r\rangle)$ we have that $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}', w \models \mathsf{Tr}(\varphi)$. □

The same expressivity diagram shown for the $\mathcal{ML}_\emptyset$ logics can be established for the logics with arbitrary memory, following equivalent arguments:

3.2.19. Theorem.

1. *$\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ over the signature $\langle \text{PROP}, \text{REL}\rangle$ is strictly more expressive than $\mathcal{ML}$ over the signature $\langle \text{PROP} \cup \{known\}, \text{REL}\rangle$.*

2. *$\mathcal{ML}^m(\langle\langle r\rangle\rangle) < \mathcal{ML}^m(\langle r\rangle)$.*

3. *$\mathcal{HL}(\downarrow)$ over the signature $\langle \text{PROP} \cup \{known\}, \text{REL}, \text{NOM}\rangle$ is strictly more expressive than $\mathcal{ML}^m(\langle r\rangle)$ over the signature $\langle \text{PROP}, \text{REL}\rangle$.*

Proof. The proof for (i) is the same as the proof for Theorem 3.2.3. The proof for (ii) is the same as the proof for Theorem 3.2.2. To prove (iii) we adapt the translation Tr defined in the proof of Theorem 3.2.4 with the following clause for ⓚ:

$$\mathsf{Tr}_N(ⓚ) \;=\; \left(\bigvee_{i\in N} i\right) \vee known.$$

$\mathcal{HL}(\downarrow) \not\leq \mathcal{ML}^m(\langle r\rangle)$ can be shown using the following models. Let $\mathcal{M}_1 = \langle\{w\}, \{(w,w)\}, \emptyset, \{w\}\rangle$ and $\mathcal{M}_2 = \langle\{u,v\}, \{(u,v),(v,u)\}, \emptyset, \{u,v\}\rangle$. Duplicator always wins on $EF(\mathcal{M}_1, \mathcal{M}_2, w, u)$ and thus $\mathcal{M}_1, w \equiv^{EF}_{\mathcal{ML}^m(\langle r\rangle)} \mathcal{M}_2, u$. On the other hand, $\mathcal{M}_1, w \models \downarrow i.\langle r\rangle i$ but $\mathcal{M}_2, u \not\models \downarrow i.\langle r\rangle i$. □

3.3 A general picture

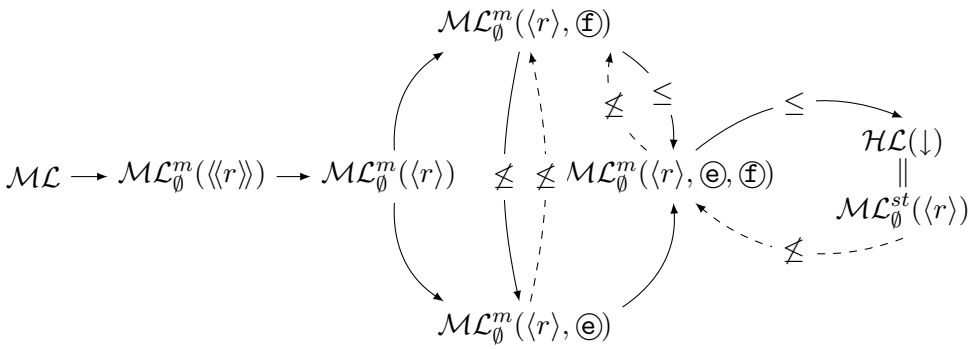
As we said before, we have not explored all possible fragments exhaustively. We tested different ingredients in the context of memory logics and we analyzed through bisimulation the expressive power of several fragments, investigating different ways in which information can be stored and retrieved.

We mainly centered our work in using sets as the information container, but we also studied the possibility of replacing the set with a "richer" structure, like the stack. Given that we worked mostly with sets, we studied the operations that one can naturally expect in this context: to add, test membership and delete elements from sets. We also distinguished the case in which we start evaluating a formula in an empty memory from the case where the storage container can come with 'a priori' information. This distinction becomes quite natural when working with models with state, in which starting from a "clean" configuration is a usual need.

We have not mentioned the case of hybrid memory logics, but the comparisons we already established among the modal memory fragments can be easily transfered to case when we have nominals and the satisfaction operator @.

We have proved that, in terms of expressive power, the memory logics we presented lie between the basic modal logic $\mathcal{K}$ and the hybrid logic $\mathcal{HL}(\downarrow)$. This allows to transfer known results from $\mathcal{HL}(\downarrow)$. For example, all these memory fragments have compactness and the generated submodel property (as $\mathcal{HL}(\downarrow)$ has). Also, we can compose the translations we defined for each memory logic with the standard translation from $\mathcal{HL}(\downarrow)$ to first order logic (see [BdRV01] for details) and build a standard translation that takes memory logics formulas to first order logic formulas.

Furthermore, we observe that structures richer than a set, although being still quite basic (like a stack, a list, etc.), give us already the expressive power of $\mathcal{HL}(\downarrow)$. The following picture summarizes the results we found in terms of expressivity. We have only included in the picture the memory fragments defined for the class $\mathcal{C}_\emptyset$ since as we said before, these fragments seem the most reasonable ones to compare between $\mathcal{K}$ and $\mathcal{HL}(\downarrow)$.



The solid unlabeled arrows represent the $<$ relationship, that is, $\mathcal{L} \to \mathcal{L}'$ means that the logic $\mathcal{L}$ is strictly less expressive than the logic $\mathcal{L}'$. In some cases we specifically indicate other relations (like $\leq$ or $\not\leq$), and the dashed arrows show the suspected answers to the open questions 3.2.15 and 3.2.16 we have formulated.

Chapter 4

(Un)Decidability

She both advanced toward him and retreated, a difficult maneuver which Mrs Frick alone could carry off. It had taken her ten decades of practice.

"Ubik", Philip K. Dick.

In Chapter 3 we showed some fragments which are all more expressive than $\mathcal{ML}$ but less expressive than $\mathcal{HL}(\downarrow)$ (with the exception of $\mathcal{ML}^{st}_{\emptyset}(\langle r \rangle)$, which is equally expressive as $\mathcal{HL}(\downarrow)$). Given that $\mathcal{ML}$ is decidable and $\mathcal{HL}(\downarrow)$ undecidable [AtC06], exploring where the decidability line lies is an intriguing question. The main goal of this chapter is to investigate this issue, together with the related question of whether the logic is sufficiently expressive to force infinite models. To study the decidability of a logic, we are going to use the techniques we outlined in Section 2.2.2.

We start by investigating $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ and $\mathcal{ML}^m_{\emptyset}(\langle\langle r \rangle\rangle)$, the weakest memory fragments we defined. We will show that the satisfiability problem for $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ is decidable (actually PSPACE-complete) while $\mathcal{ML}^m_{\emptyset}(\langle\langle r \rangle\rangle)$ is already undecidable. As we will show in this chapter, the trick to prove decidability is to unravel a model use a 'dirty' memory. On the other hand, in $\mathcal{ML}^m_{\emptyset}(\langle\langle r \rangle\rangle)$, we are restricted to the class of models where the memory starts empty, and we can't play this trick anymore. Actually $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ is really standing on the decidability line: adding a single nominal to $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$, or restricting the class of models to $\mathcal{C}_{\emptyset}$ pushes the satisfiability problem over to undecidability. This will help us to establish a general result that includes several memory fragments.

For all the undecidable fragments we analyze, we are also going to prove that they lack the finite model property. This is not necessarily always true, but in a given logic these two properties are usually present together.

4.1 The decidability of $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$

We will first prove that $\mathcal{ML}$ and $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ are expressively equivalent over the class of tree-like models. We will then prove that $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ has the *tree model property*, that is to say, if a formula $\varphi \in \mathcal{ML}^m(\langle\langle r \rangle\rangle)$ is satisfiable, then it is satisfiable in a tree-like model. With those results at hand, decidability and PSPACE-completeness of $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ easily follows.

4.1.1. THEOREM. *The logic $\mathcal{ML}$ over the signature $\langle \text{PROP} \cup \{known\}, \text{REL}\rangle$ is equivalent to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ over the class of tree models.*

PROOF. [$\mathcal{ML} \leq \mathcal{ML}^m(\langle\langle r\rangle\rangle)$]: We can take the translation that maps *known* to ⓚ, $\langle r\rangle$ to $\langle\langle r\rangle\rangle$, and it is the identity for the rest of the operators. In a similar way we did in Theorem 3.2.18, we associate every model $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V\rangle$ of the logic $\mathcal{ML}$ over $\langle \text{PROP}\cup\{known\}, \text{REL}\rangle$ with the model $\mathcal{M}' = \langle W, (R_r)_{r\in\text{REL}}, V', S\rangle$ of the logic $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ over $\langle \text{PROP}, \text{REL}\rangle$ where V' is identical to V over PROP and $S = V(known)$. Since we are in the class of tree-like models, it is easy to see that this translation is equivalence preserving.

[$\mathcal{ML}^m(\langle\langle r\rangle\rangle) \leq \mathcal{ML}$]: We start by noticing that in $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ we can eliminate ⓚ at modal depth zero from a formula like ⓡφ.

4.1.2. CLAIM. Let $\varphi^\sharp$ be the result of replacing all the occurrences of ⓚ that are in $\varphi \in \mathcal{ML}^m(\langle\langle r\rangle\rangle)$ at modal depth zero by $\top$. Then $\mathcal{M}, w \models$ ⓡφ iff $\mathcal{M}, w \models \varphi^\sharp$.

PROOF. We proceed by induction on φ. The case for ⓚ, the propositional symbols and booleans are straightforward. We analyze the other cases:

- $\varphi =$ ⓡψ. $\mathcal{M}, w \models$ ⓡⓡψ iff $\mathcal{M}, w \models$ ⓡψ iff (by inductive hypothesis) $\mathcal{M}, w \models \psi^\sharp$ iff $\mathcal{M}, w \models (\psi^\sharp)^\sharp$ iff (by inductive hypothesis) $\mathcal{M}, w \models$ ⓡ$(\psi^\sharp)$ iff $\mathcal{M}, w \models$ (ⓡ$\psi)^\sharp$.
- $\varphi = \langle\langle r\rangle\rangle\psi$. $\mathcal{M}, w \models$ ⓡ$\langle\langle r\rangle\rangle\psi$ iff (by definition) $\mathcal{M}[w], w \models \langle\langle r\rangle\rangle\psi$ iff (by definition of $\sharp$) $\mathcal{M}[w], w \models (\langle\langle r\rangle\rangle\psi)^\sharp$ iff (by definition) $\mathcal{M}, w \models (\langle\langle r\rangle\rangle\psi)^\sharp$.

□

We define now the following translation, taking $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas over the signature $\langle \text{PROP}, \text{REL}\rangle$ to $\mathcal{ML}$-formulas over the signature $\langle \text{PROP}\cup\{known\}, \text{REL}\rangle$:

$$\begin{array}{rcl}
\mathsf{Tr}(p) & = & p \quad p \in \text{PROP} \\
\mathsf{Tr}(\text{ⓚ}) & = & known \\
\mathsf{Tr}(\neg\varphi) & = & \neg\mathsf{Tr}(\varphi) \\
\mathsf{Tr}(\varphi_1 \wedge \varphi_2) & = & \mathsf{Tr}(\varphi_1) \wedge \mathsf{Tr}(\varphi_2) \\
\mathsf{Tr}(\langle\langle r\rangle\rangle\varphi) & = & \langle r\rangle\mathsf{Tr}(\varphi) \\
\mathsf{Tr}(\text{ⓡ}\varphi) & = & \mathsf{Tr}(\varphi^\sharp)
\end{array}$$

Let $\varphi \in \mathcal{ML}^m(\langle\langle r\rangle\rangle)$, and let $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ be an arbitrary tree model. Let $\mathcal{M}' = \langle W, (R_r)_{r\in\text{REL}}, V'\rangle$ where V' is identical to V except that $V'(known) = S$. We can prove that $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}', w \models \mathsf{Tr}(\varphi)$.

We proceed by induction on φ. The propositional and boolean cases are trivial. The ⓚ case is also easy given the definitions. Let us consider $\varphi = \langle\langle r\rangle\rangle\psi$. Given that $\mathcal{M}$ is tree-like, the remember operator has no effect beyond modal operators, so $\mathcal{M}, w \models \langle\langle r\rangle\rangle\psi$ iff exists v such that $R(w, v)$ and $\mathcal{M}, v \models \psi$. By inductive

hypothesis, $\mathcal{M}', v \models \psi$ iff $\mathcal{M}', v \models \mathsf{Tr}(\psi)$, and by definition $\mathcal{M}', w \models \langle r\rangle\mathsf{Tr}(\psi)$. Finally, let us see the case for remember. By Claim 4.1.2, $\mathcal{M}, w \models \textcircled{r}\psi$ iff $\mathcal{M}, w \models \psi^\sharp$. By inductive hypothesis, $\mathcal{M}, w \models \mathsf{Tr}(\psi^\sharp)$. □

We now prove that $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ has the tree model property. The idea is to use a similar technique than the one used for the basic modal logic (see [BdRV01]), in which each path in the original model is represented by a different point in the tree-like model. The "already visited" points are simulated in the tree model by points that are initially stored in the memory. The exact procedure to transform a model in a tree-like equivalent model will be clear from the proof of Theorem 4.1.3, but let's see first an example:

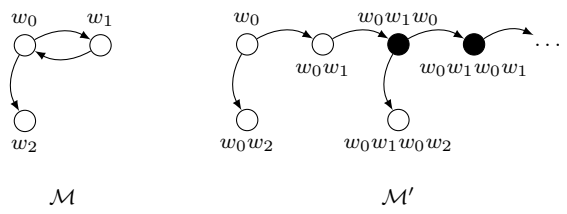


Figure 4.1: A model $\mathcal{M}$ and its corresponding tree-like model $\mathcal{M}'$

Figure 4.1 shows a model $\mathcal{M}$ and the resulting tree-like model $\mathcal{M}'$, taking w_0 as the starting point. The model $\mathcal{M}'$ is essentially the unraveling of $\mathcal{M}$, in which every point represents a path starting from w_0 in the original model. But additionally, the procedure takes into consideration the points that are memorized during the exploration of the model, and these points are added to the initial memory of $\mathcal{M}'$. The idea is to add to the initial memory the points that represent paths whose last point is an already visited point. In the example above, that is the case for $w_0w_1w_0$ since w_0 is visited twice there, but it is not for $w_0w_1w_0w_2$, since w_2 is visited just once.

4.1.3. Theorem (Tree model property). *Let $\mathcal{M}$ be an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-model, $w \in \mathcal{M}$. Then there is a tree-like model $\mathcal{M}'$ such that $\langle \mathcal{M}, w\rangle \equiv^{EF} \langle \mathcal{M}', w\rangle$.*

Proof. We prove the result for the unimodal case, the generalization to the multimodal case is straightforward. Let $\mathcal{M} = \langle W, R, V, S\rangle$, define $\mathcal{M}' = \langle W', R', V', S'\rangle$ as follows. Its domain W' consists of all finite sequences $(u_0, \ldots, u_n)$ such that $u_0 = w$, $n \geq 0$ and there is a path $u_0Ru_1 \ldots Ru_n$ in $\mathcal{M}$. Define $(u_0, \ldots, u_n)R(v_0, \ldots, v_m)$ to hold if $m = n + 1$, $u_i = v_1$ for $i = 0, \ldots, n$ and u_nRv_m holds in $\mathcal{M}$. The valuation V' is defined by setting $(u_0, \ldots, u_n) \in V'(p)$ iff $u_n \in V(p)$. Finally, $(u_0, \ldots, u_{n-1}, u_n) \in S'$ iff $u_n \in \{u_0, \ldots, u_{n-1}\}$ or $u_n \in S$.

Let s_i be the sequence $(v_0, \ldots, v_i)$. We show that Duplicator has a winning strategy in the game $EF(\mathcal{M}, \mathcal{M}', w, w)$. It is sufficient to see that in the game $EF(\mathcal{M}[v_0, \ldots, v_n], \mathcal{M}'[s_0, \ldots, s_n], v_{n+1}, s_{n+1})$, Duplicator can always answer successfully to Spoiler's moves.

- If Spoiler chooses $\mathcal{M}[v_0, \ldots, v_n]$ and some v_{n+1}, a successor of v_n, Duplicator chooses the sequence $s_{n+1} = s_n v_{n+1}$.

- If Spoiler chooses $\mathcal{M}'[s_0, \ldots, s_n]$ and $s_{n+1} = s_n v_{n+1}$ (for some v_{n+1}), a successor of s_n, Duplicator chooses the point v_{n+1}.

By definition s_{n+1} and v_{n+1} agree. Observe that the memory of $\mathcal{M}[v_0, \ldots, v_n]$ is $S \cup \{v_0, \ldots, v_n\}$ and the memory of $\mathcal{M}'[s_0, \ldots, s_n]$ is $S' \cup \{s_0, \ldots, s_n\}$. It is also clear that $v_{n+1} \in S$ iff $s_{n+1} \in S'$. Formally, $v_{n+1} \in S \cup \{v_0, \ldots, v_n\}$ implies $s_{n+1} \in S'$ by definition. And $s_{n+1} \in S' \cup \{s_0, \ldots, s_n\}$, then $s_{n+1} \in S'$ (since there are no cycles in $\mathcal{M}'$) and by definition $v_{n+1} \in S \cup \{v_0, \ldots, v_n\}$. $\square$

Observe that having the possibility to use a nonempty memory is crucial to construct the tree-like model. As we are going to see later, when we restrict the logic to the class $\mathcal{C}_\emptyset$, not only the tree model property does not hold anymore, but we cross the border of decidability and the resulting logic is undecidable.

We have just proved that $\mathcal{ML}$ and $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ are equivalent over the class of tree-like models and that $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ has the tree model property. Given that we already know that $\mathcal{ML}$ also has the tree model property, the expected conclusion would be that $\mathcal{ML}$ and $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ are equivalent logics. On the other hand, although we have not shown this explicitly in Chapter 3, $\mathcal{ML} < \mathcal{ML}^m(\langle\langle r \rangle\rangle)$ is a result that can be easily transfered from Theorem 3.2.3: the two models shown there are $\mathcal{ML}$-bisimilar, and distinguishable with the $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$-formula $\langle\langle r \rangle\rangle ⓚ$. Where is the contradiction then? The answer is that the two logics are equivalent over the class of tree models *using a shift in the signature.* Adding the propositional symbol *known* to the signature of $\mathcal{ML}$ is crucial to establish the equivalence, since it enables the ability to encode the memorized points. This *does not* mean that the two logics are equivalent in the general sense.

With the results shown in Theorems 4.1.3 and 4.1.1, the following theorem easily follows.

4.1.4. Theorem. *The satisfiability problem of $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ is* PSPACE*-complete.*

Proof. We first show the decidability of the satisfiability problem of $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ proving that any satisfiable formula of $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ is satisfiable in a recursively bounded model.

Let φ be an $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$-formula of degree k, and suppose $\mathcal{M}_1, w \models \varphi$. By Theorem 4.1.3, there is a tree-like model $\mathcal{M}_2$ such that $\mathcal{M}_2, w \models \varphi$. Using Theorem 4.1.1, we know that $\mathcal{M}_2, w \models \mathsf{Tr}(\varphi)$ (here, $\mathcal{M}_2$ is taken as an $\mathcal{ML}$ model over the appropriate signature). Now we can use the *bounded* tree model property for basic modal logic [BdRV01], so there must be a recursively bounded tree-like model $\mathcal{M}_3 = \langle W, (R_r)_{r \in \text{REL}}, V \rangle$ and $v \in W$ such that $\mathcal{M}_3, v \models \mathsf{Tr}(\varphi)$. Finally, we can use Theorem 4.1.1 again, and conclude $\langle W, (R_r)_{r \in \text{REL}}, V, V(\mathit{known}) \rangle, v \models \varphi$.

The PSPACE-completeness follows from the fact that the translation Tr is linear, and that the satisfiability problem for the basic modal logic is PSPACE (refer to [BdRV01]). □

4.1.1 A linear encoding for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$

Now we show that adding $\textcircled{f}$ to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ keeps the logic decidable. In fact, we will show that $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ can be encoded into $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ using a linear translation. We are going to see later in this chapter that this is not the case for $\textcircled{e}$.

4.1.5. THEOREM. *The satisfiability problem for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ is* PSPACE-*complete.*

PROOF. We show that there is a linear equivalence preserving translation that takes formulas from $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. Let Tr_a be the following translation from $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ formulas to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ formulas, where a ranges over $\{r, f\}$:

$$\begin{array}{rcl}
\mathsf{Tr}_a(p) & = & p \quad p \in \text{PROP} \\
\mathsf{Tr}_r(\textcircled{k}) & = & \textcircled{k} \\
\mathsf{Tr}_f(\textcircled{k}) & = & \neg\top \\
\mathsf{Tr}_a(\neg\varphi) & = & \neg\mathsf{Tr}_a(\varphi) \\
\mathsf{Tr}_a(\varphi_1 \wedge \varphi_2) & = & \mathsf{Tr}_a(\varphi_1) \wedge \mathsf{Tr}_a(\varphi_2) \\
\mathsf{Tr}_a(\langle\langle r\rangle\rangle\varphi) & = & \langle\langle r\rangle\rangle\mathsf{Tr}_r(\varphi) \\
\mathsf{Tr}_a(\textcircled{r}\varphi) & = & \textcircled{r}\mathsf{Tr}_r(\varphi) \\
\mathsf{Tr}_a(\textcircled{f}\varphi) & = & \mathsf{Tr}_f(\varphi)
\end{array}$$

We prove by mutual induction on φ these two properties:

1. $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}, w \models \mathsf{Tr}_r(\varphi)$.
2. $\mathcal{M}[-w], w \models \varphi$ iff $\mathcal{M}, w \models \mathsf{Tr}_f(\varphi)$.

The only interesting cases for both properties are $\langle\langle r\rangle\rangle, \textcircled{r}$ and $\textcircled{f}$. For the property (i), let $\varphi = \langle\langle r\rangle\rangle\psi$. $\mathcal{M}, w \models \langle\langle r\rangle\rangle\psi$ iff (by definition) there is a $w' \in W$, such that wRw' and $\mathcal{M}[w], w' \models \psi$ iff (by inductive hypothesis on (i)) $\mathcal{M}[w], w' \models \mathsf{Tr}_r(\psi)$ iff (by definition) $\mathcal{M}, w \models \langle\langle r\rangle\rangle\mathsf{Tr}_r(\psi)$ iff (by definition) $\mathcal{M}, w \models \mathsf{Tr}_r(\langle\langle r\rangle\rangle\psi)$. The next case is $\varphi = \textcircled{r}\psi$. $\mathcal{M}, w \models \textcircled{r}\psi$ iff (by definition) $\mathcal{M}[w], w \models \psi$ iff (by inductive hypothesis on (i)) $\mathcal{M}[w], w \models \mathsf{Tr}_r(\psi)$ iff (by definition) $\mathcal{M}, w \models \textcircled{r}\mathsf{Tr}_r(\psi)$ iff (by definition) $\mathcal{M}, w \models \mathsf{Tr}_r(\textcircled{r}\psi)$. Finally, let $\varphi = \textcircled{f}\psi$. $\mathcal{M}, w \models \textcircled{f}\psi$ iff (by definition) $\mathcal{M}[-w], w \models \psi$ iff (by inductive hypothesis on (ii)) $\mathcal{M}, w \models \mathsf{Tr}_f(\psi)$ iff (by definition) $\mathcal{M}, w \models \mathsf{Tr}_r(\textcircled{f}\psi)$.

For the property (ii), let $\varphi = \langle\langle r\rangle\rangle\psi$. $\mathcal{M}[-w], w \models \langle\langle r\rangle\rangle\psi$ iff (by definition) there is a $w' \in W$, such that wRw' and $\mathcal{M}[w], w' \models \psi$ iff (by inductive hypothesis

on (i)) $\mathcal{M}[w], w' \models \mathsf{Tr}_r(\psi)$ iff (by definition) $\mathcal{M}, w \models \langle\langle r \rangle\rangle \mathsf{Tr}_r(\psi)$ iff (by definition) $\mathcal{M}, w \models \mathsf{Tr}_f(\langle\langle r \rangle\rangle \psi)$. The next case is $\varphi = \textcircled{r}\psi$. $\mathcal{M}[-w], w \models \textcircled{r}\psi$ iff (by definition) $\mathcal{M}[w], w \models \psi$ (by inductive hypothesis on (i)) $\mathcal{M}[w], w \models \mathsf{Tr}_r(\psi)$ iff (by definition) $\mathcal{M}, w \models \textcircled{r}\mathsf{Tr}_r(\psi)$ iff (by definition) $\mathcal{M}, w \models \mathsf{Tr}_f(\textcircled{r}\psi)$. The last case is $\varphi = \textcircled{f}\psi$. $\mathcal{M}[-w], w \models \textcircled{f}\psi$ iff (by definition) $\mathcal{M}[-w], w \models \psi$ (by inductive hypothesis on (ii)) $\mathcal{M}, w \models \mathsf{Tr}_f(\psi)$ iff (by definition) $\mathcal{M}, w \models \mathsf{Tr}_f(\textcircled{f}\psi)$.

Therefore, property (i) shows that we have defined an equivalent preserving translation. Observe that the linearity of the translation is trivial, and given Theorem 4.1.4, we conclude the desired result. □

Given the above theorem, the tree model property for $\mathcal{ML}^m(\langle\langle r \rangle\rangle, \textcircled{f})$ easily follows:

4.1.6. Corollary. *Let φ be a formula of $\mathcal{ML}^m(\langle\langle r \rangle\rangle, \textcircled{f})$. If φ is satisfiable, then it is satisfied in a tree-like model.*

Proof. Let $\mathcal{M}$ and $w \in \mathcal{M}$ be such that $\mathcal{M}, w \models \varphi$. We can use the translation defined in the proof of the above theorem, and therefore $\mathcal{M}, w \models \mathsf{Tr}_r(\varphi)$, where $\mathsf{Tr}_r(\varphi)$ is a formula of $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$. Using Theorem 4.1.3, there is a tree-like model $\mathcal{M}'$ such that $\mathcal{M}', w \models \mathsf{Tr}_r(\varphi)$. Given that tr_r is an equivalence preserving translation, $\mathcal{M}', w \models \varphi$. □

4.2 Some undecidable neighbors of $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$

While $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ is decidable, it seems to be standing at the border of undecidability. The logic $\mathcal{ML}^m_\emptyset(\langle\langle r \rangle\rangle)$, obtained from $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ by restricting the models to the class $\mathcal{C}_\emptyset$, is undecidable. Furthermore, if we take $\mathcal{HL}^m(\langle\langle r \rangle\rangle)$ ($\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ with nominals), even if we restrict the signature to just one nominal, the resulting logic is also undecidable. To prove undecidability we use the tiling technique presented in Section 2.2.2.

4.2.1 Just one nominal is enough

First we are going to analyze the case of $\mathcal{HL}^m(\langle\langle r \rangle\rangle)$ with just one nominal. We call this logic $\mathcal{HL}^m_1(\langle\langle r \rangle\rangle)$. We start by showing the failure of the finite model property.

4.2.1. Theorem. *$\mathcal{HL}^m_1(\langle\langle r \rangle\rangle)$ lacks the finite model property.*

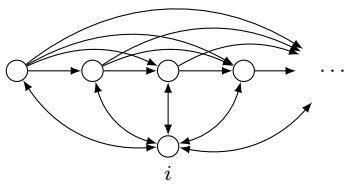
PROOF. Let's suppose that the only nominal in NOM is i. Consider the following formulas:

$$
\begin{array}{rl}
(\mathit{Back}) & i \wedge [\![r]\!]\neg i \wedge \langle\langle r\rangle\rangle\top \wedge [\![r]\!]\langle\langle r\rangle\rangle i \\
(\mathit{Empty}) & [\![r]\!]\neg ⓚ \wedge [\![r]\!][\![r]\!](\neg i \to \neg ⓚ) \\
(\mathit{Spy}) & [\![r]\!][\![r]\!](\neg i \to \langle\langle r\rangle\rangle(i \wedge \langle\langle r\rangle\rangle(ⓚ \wedge \neg\langle\langle r\rangle\rangle(ⓚ \wedge \neg i)))) \\
(\mathit{Succ}) & [\![r]\!]\langle\langle r\rangle\rangle\neg i \\
(\mathit{No\text{-}3cyc}) & \neg\langle\langle r\rangle\rangle\langle\langle r\rangle\rangle(\neg ⓚ \wedge \langle\langle r\rangle\rangle(\neg ⓚ \wedge \langle\langle r\rangle\rangle(ⓚ \wedge \neg i))) \\
(\mathit{Tran}) & [\![r]\!]\langle\langle r\rangle\rangle(i \wedge [\![r]\!](\neg ⓚ \to \langle\langle r\rangle\rangle(i \wedge \langle\langle r\rangle\rangle(ⓚ \wedge \langle\langle r\rangle\rangle(ⓚ \wedge \neg i))))).
\end{array}
$$

Let *Inf* be *Back*$\wedge$ *Empty*$\wedge$ *Spy*$\wedge$ *Succ*$\wedge$ *No-3cyc*$\wedge$ *Tran*. Let $\mathcal{M} = \langle W, R, V, S\rangle$. We claim that if $\mathcal{M}, w \models \mathit{Inf}$, then W is infinite.

Suppose $\mathcal{M}, w \models \mathit{Inf}$. Let $B = \{b \in W \mid R(w, b)\}$. Because *Back* is satisfied, $w \notin B$, $B \neq \emptyset$ and for all $b \in B$, $R(b, w)$. Note that *Empty* says that the one and two-step neighbors of w are not in S, and also this implies that every point in B is irreflexive. Because *Spy* is satisfied, if $a \neq w$ and a is a successor of an element of B then a is also an element of B. As *Succ* is satisfied at w, every point in B has a successor distinct from w. *No-3cyc* disallow cycles of size 2 or 3 in B; and together with *Tran* force R to transitively order B.

It follows that B is an unbounded strict partial order as showed in the picture below, hence infinite, and so is W.



□

We now show that $\mathcal{HL}_1^m(\langle\langle r\rangle\rangle)$ is undecidable by encoding the $\omega \times \omega$ tiling problem. Following the idea in [BS95], we will construct a *spy point* over the relation R_s; that is, the point of evaluation will have access in one R_s-step to any reachable point in the model. The relations R_u and R_r represent moving up and to the right, respectively, from one tile to the other. We code each type of tile with a fixed propositional symbol t_i. With this encoding we define for each tiling problem T, a formula φ^T such that the set of tiles T tiles $\omega \times \omega$ iff φ^T has a model.

4.2.2. THEOREM. *The satisfiability problem for $\mathcal{HL}_1^m(\langle\langle r\rangle\rangle)$ is undecidable.*

PROOF. Let $T = \{T_1, \ldots, T_n\}$ be a set of tile types. Given a tile type T_i, $u(T_i)$, $r(T_i)$, $d(T_i)$, $l(T_i)$ will represent the colors of the up, right, down and left edges

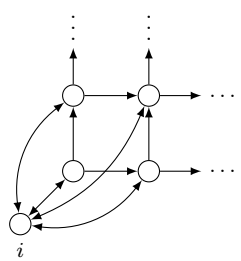
of T_i respectively. Let's suppose that i is the only nominal in NOM. Define:

$$
\begin{array}{rl}
(\mathit{Back}) & i \wedge [\![s]\!]\neg i \wedge \langle\!\langle s\rangle\!\rangle\top \wedge [\![s]\!]\langle\!\langle s\rangle\!\rangle i \wedge [\![s]\!][\![s]\!]i \\
(\mathit{Empty}) & [\![s]\!]\neg ⓚ \wedge [\![s]\!][\![\dagger]\!]\neg ⓚ \quad \dagger \in \{r,u\} \\
(\mathit{Spy}) & [\![s]\!][\![\dagger]\!]\langle\!\langle s\rangle\!\rangle(i \wedge \langle\!\langle s\rangle\!\rangle(ⓚ \wedge \neg\langle\!\langle \dagger\rangle\!\rangle ⓚ)) \quad \dagger \in \{r,u\} \\
(\mathit{Grid}) & [\![s]\!]\langle\!\langle \dagger\rangle\!\rangle\top \quad \dagger \in \{r,u\} \\
(\mathit{Func}) & [\![s]\!][\![\dagger]\!]\langle\!\langle s\rangle\!\rangle\langle\!\langle s\rangle\!\rangle(ⓚ \wedge \langle\!\langle \dagger\rangle\!\rangle ⓚ \wedge [\![\dagger]\!]ⓚ) \quad \dagger \in \{r,u\} \\
(\mathit{Conf}) & [\![s]\!]\langle\!\langle u\rangle\!\rangle\langle\!\langle r\rangle\!\rangle\langle\!\langle s\rangle\!\rangle\langle\!\langle s\rangle\!\rangle(ⓚ \wedge \neg\langle\!\langle r\rangle\!\rangle ⓚ \wedge \langle\!\langle u\rangle\!\rangle ⓚ \wedge \\
 & \quad \langle\!\langle r\rangle\!\rangle(\neg ⓚ \wedge (\langle\!\langle u\rangle\!\rangle(ⓚ \wedge \neg\langle\!\langle r\rangle\!\rangle ⓚ))) \\
(\mathit{UR\text{-}no\text{-}cyc}) & [\![s]\!][\![u]\!][\![r]\!]\neg ⓚ \wedge [\![s]\!][\![r]\!][\![u]\!]\neg ⓚ \\
(\mathit{URU\text{-}no\text{-}cyc}) & [\![s]\!][\![u]\!][\![r]\!][\![u]\!]\neg ⓚ \\
(\mathit{Unique}) & [\![s]\!]\left(\bigvee_{1\le i\le n} t_i \wedge \bigwedge_{1\le i<j\le n}(t_i \to \neg t_j)\right) \\
(\mathit{Vert}) & [\![s]\!]\bigwedge_{1\le i\le n}\left(t_i \to \langle\!\langle u\rangle\!\rangle \bigvee_{1\le j\le n, u(T_i)=d(T_j)} t_j\right) \\
(\mathit{Horiz}) & [\![s]\!]\bigwedge_{1\le i\le n}\left(t_i \to \langle\!\langle r\rangle\!\rangle \bigvee_{1\le j\le n, r(T_i)=l(T_j)} t_j\right)
\end{array}
$$

Let the formula φ^T be the conjunction of all the above formulas. We show that T tiles $\omega \times \omega$ iff φ^T is satisfiable.

Suppose $\mathcal{M}, w \models \varphi^T$. Observe that *Back* and *Spy*, together with *Empty* make w a spy via R_s (and also force R_u and R_r to be irreflexive and asymmetric). These R_s-accessible points will represent the tiles. We'll have that $[\![s]\!]\psi$ holds at w iff ψ is true at every tile, and $\langle\!\langle s\rangle\!\rangle\langle\!\langle s\rangle\!\rangle\psi$ holds at tile v iff ψ is true at some (perhaps the same) tile. Now, *Grid* states that from every tile there is another tile moving up (that is, following the R_u-relation). The same holds for the right direction (following the R_r-relation). *Func* (together with *Back* and *Spy*) forces R_u and R_r to be functional. *Conf* ensures that the tiles are arranged as a grid, once we force $R_u \circ R_r$ to be irreflexive (*UR-Irr*), and we forbid cycles via the $R_u R_r R_u$ pattern (*URU-no-cyc*).

That completes the description of the grid. The last three formulas ensure that every tile has a unique type t_i, and that the colors of the tiles match properly. From this, it easily follows that $\mathcal{M}$ is a tiling of $\omega \times \omega$.



For the converse, suppose $f : \omega \times \omega \to T$ is a tiling of $\omega \times \omega$. We define the model $\mathcal{M} = \langle W, \{R_s, R_u, R_r\}, V, \emptyset\rangle$ as follows:

- $W = \omega \times \omega \cup \{w\}$
- $R_s = \{(w, v), (v, w) \mid v \in \omega \times \omega\}$ (hence w will act as the spy point)
- $R_u = \{((x, y), (x, y + 1)) \mid x, y \in \omega\}$
- $R_r = \{((x, y), (x + 1, y)) \mid x, y \in \omega\}$
- $V(p) = \{w\}$; $V(t_i) = \{x \mid x \in \omega \times \omega, f(x) = T_i\}$

The reader may verify that, by construction, $\mathcal{M}, w \models \varphi^T$. □

4.2.2 An empty memory is also enough

We now turn to the case for $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$. The ideas are similar to the case of $\mathcal{HL}^m_1(\langle\langle r\rangle\rangle)$, but this time we cannot use the nominal i to make the spy point. On the other hand, we know that the memory is empty when we start evaluating a formula, and this will help us to encode the tiling.

As we did before, we first show the failure of the finite model property.

4.2.3. Theorem. *$\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ lacks the finite model property.*

Proof. Consider the following formulas:

$$
\begin{array}{rl}
(\mathit{Back}) & p \wedge [\![r]\!]\neg p \wedge \langle\langle r\rangle\rangle\top \wedge [\![r]\!]\langle\langle r\rangle\rangle(\textcircled{k} \wedge p) \\
(\mathit{Spy}) & [\![r]\!][\![r]\!](\neg p \rightarrow \langle\langle r\rangle\rangle(\textcircled{k} \wedge p \wedge \langle\langle r\rangle\rangle(\textcircled{k} \wedge \neg\langle\langle r\rangle\rangle(\textcircled{k} \wedge \neg p)))) \\
(\mathit{Irr}) & \neg\langle\langle r\rangle\rangle(\neg p \wedge \langle\langle r\rangle\rangle(\neg p \wedge \textcircled{k}))) \\
(\mathit{Succ}) & [\![r]\!]\langle\langle r\rangle\rangle\neg p \\
(\mathit{No\text{-}3cyc}) & \neg\langle\langle r\rangle\rangle\langle\langle r\rangle\rangle(\neg\textcircled{k} \wedge \langle\langle r\rangle\rangle(\neg\textcircled{k} \wedge \langle\langle r\rangle\rangle(\textcircled{k} \wedge \neg p))) \\
(\mathit{Tran}) & [\![r]\!](\neg p \rightarrow \langle\langle r\rangle\rangle(\textcircled{k} \wedge p \wedge [\![r]\!](\neg p \wedge \neg\textcircled{k} \rightarrow \langle\langle r\rangle\rangle(\textcircled{k} \wedge p \wedge \\
 & \quad \langle\langle r\rangle\rangle(\textcircled{k} \wedge \neg p \wedge \langle\langle r\rangle\rangle(\textcircled{k} \wedge \neg p))))))
\end{array}
$$

Let *Inf* be *Back* $\wedge$ *Spy* $\wedge$ *Irr* $\wedge$ *Succ* $\wedge$ *No-3cyc* $\wedge$ *Tran*, and let $\mathcal{M} = \langle W, R, V, \emptyset\rangle$. The proof that $\mathcal{M}$ is infinite if $\mathcal{M}, w \models$ *Inf* is similar to the proof of Theorem 4.2.1. Instead of using i to identify the spy point we now use p and $\textcircled{k}$. $\textcircled{k}$ is needed to distinguish the spy point from other points where p might hold.

Notice that *Back*, *Spy*, *Succ*, and *No-3cyc* are very similar to the ones in the proof of Theorem 4.2.1, *Irr* forces R to be irreflexive and *Tran* says that every pair of successors u and v are related (either $R(u, v)$ or $R(v, u)$), and this together with the other formulas, implies that R is transitive.

□

In a similar way, we can encode the $\omega \times \omega$ tiling problem to show that satisfiability in $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ is undecidable.

4.2.4. Theorem. *The satisfiability problem for $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ is undecidable.*

PROOF. The formula φ^T needed for the encoding of a tiling problem T in this case is the conjunction of the following:

$$
\begin{array}{rl}
(\mathit{Back}) & p \wedge [\![s]\!]\neg p \wedge \langle\!\langle s\rangle\!\rangle\top \wedge [\![s]\!]\langle\!\langle s\rangle\!\rangle(\textcircled{k} \wedge p) \wedge [\![s]\!][\![s]\!](\textcircled{k} \wedge p) \\
(\mathit{Spy}) & [\![s]\!][\![\dagger]\!](\neg p \wedge \langle\!\langle s\rangle\!\rangle(\textcircled{k} \wedge p \wedge \langle\!\langle s\rangle\!\rangle(\textcircled{k} \wedge \neg\langle\!\langle\dagger\rangle\!\rangle\textcircled{k}))) \quad \dagger \in \{r, u\} \\
(\mathit{Grid}) & [\![s]\!]\langle\!\langle\dagger\rangle\!\rangle\top \quad \dagger \in \{r, u\} \\
(\mathit{Func}) & [\![s]\!][\![\dagger]\!]\langle\!\langle s\rangle\!\rangle\langle\!\langle s\rangle\!\rangle(\textcircled{k} \wedge \langle\!\langle\dagger\rangle\!\rangle\textcircled{k} \wedge [\![\dagger]\!]\textcircled{k}) \quad \dagger \in \{r, u\} \\
(\mathit{Conf}) & [\![s]\!]\langle\!\langle u\rangle\!\rangle\langle\!\langle r\rangle\!\rangle\langle\!\langle s\rangle\!\rangle\langle\!\langle s\rangle\!\rangle(\textcircled{k} \wedge \neg\langle\!\langle r\rangle\!\rangle\textcircled{k} \wedge \langle\!\langle u\rangle\!\rangle\textcircled{k} \wedge \\
 & \quad \langle\!\langle r\rangle\!\rangle\langle\!\langle u\rangle\!\rangle(\textcircled{k} \wedge \neg\langle\!\langle r\rangle\!\rangle\textcircled{k})) \\
(\mathit{URU\text{-}no\text{-}cyc}) & [\![s]\!][\![u]\!][\![r]\!]\neg\textcircled{k} \wedge [\![s]\!][\![r]\!][\![u]\!]\neg\textcircled{k} \\
(\mathit{URU\text{-}no\text{-}cyc}) & [\![s]\!][\![u]\!][\![r]\!][\![u]\!]\neg\textcircled{k} \\
(\mathit{Unique}) & [\![s]\!]\left(\bigvee_{1\leq i\leq n} t_i \wedge \bigwedge_{1\leq i<j\leq n}(t_i \rightarrow \neg t_j)\right) \\
(\mathit{Vert}) & [\![s]\!]\bigwedge_{1\leq i\leq n}\left(t_i \rightarrow \langle\!\langle u\rangle\!\rangle \bigvee_{1\leq j\leq n, u(T_i)=d(T_j)} t_j\right) \\
(\mathit{Horiz}) & [\![s]\!]\bigwedge_{1\leq i\leq n}\left(t_i \rightarrow \langle\!\langle r\rangle\!\rangle \bigvee_{1\leq j\leq n, r(T_i)=l(T_j)} t_j\right)
\end{array}
$$

□

Now we turn to the case of $\mathcal{ML}^m(\langle r\rangle)$, making use of the results we just presented. To prove failure of the finite model property for the case $\mathcal{ML}^m(\langle r\rangle)$, observe that the following lemma is easy to establish (we only state it for the mono-modal case; a similar result is true in the multimodal case). The idea is that we can write a formula that forces a model to have a memory with a "clean horizon" of size d from a given point, and then reuse the encoding we defined for $\mathcal{ML}^m_\emptyset(\langle\!\langle r\rangle\!\rangle)$. Failure of the finite model property is then a direct consequence.

4.2.5. LEMMA. *Let φ be a formula of modal depth d. If $\langle W, R_r, V, S\rangle, w \models \left(\bigwedge_{i=0}^{d}[r]^i\neg\textcircled{k}\right) \wedge \varphi$ then $\langle W, R_r, V, \emptyset\rangle, w \models \varphi$.*

4.2.6. COROLLARY. *$\mathcal{ML}^m(\langle r\rangle)$ lacks the finite model property.*

PROOF. Using Lemma 4.2.5, one can easily see that the formula $\left(\bigwedge_{i=0}^{4}[r]^i\neg\textcircled{k}\right) \wedge$ *Inf*, where *Inf* is the one in the proof of Theorem 4.2.3, forces an infinite model. □

4.2.7. COROLLARY. *The satisfiability problem for $\mathcal{ML}^m(\langle r\rangle)$ is undecidable.*

PROOF. Using the idea of Lemma 4.2.5 and the formula φ^T in the proof of Theorem 4.2.4, we can obtain a formula ψ such that if $\mathcal{M}, w \models \psi$ then $\mathcal{M}$ is a tiling of $\omega \times \omega$. For the converse, we can build exactly the same model as in the proof of Theorem 4.2.4 and check that it satisfies ψ. □

Now we want to briefly mention the case of erase with respect to decidability. Given that ⓔ can be seen as an operator that internalizes the notion of starting the evaluation of a formula with an empty memory, it is quite easy to establish the following result:

4.2.8. THEOREM. *The satisfiability problem for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓔ)$ is undecidable. Furthermore, the logic lacks the finite model property.*

PROOF. Theorems 4.2.3 and 4.2.4 show that the logic $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ lacks the finite model property and that its satisfiability problem is undecidable. It is straightforward to see that just adding ⓔ in front of each encoding is enough to achieve the same results for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓔ)$. □

Summing up, $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ and $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ are decidable (in fact, they are PSPACE-complete). In contrast, from the results shown in Theorems 4.2.2, 4.2.4, 4.2.8 and Corollary 4.2.7, the undecidability of the remaining memory logics we are studying easily follows.

Chapter 5

Interpolation and Beth definability

Basta, de hablar,
acuso que el lenguaje humano
desde siempre es inútil
y está todo mal.

"La logia", Bersuit Vergarabat.

In the last chapters we presented some results regarding expressivity and decidability for several memory logics. Now it is the turn to analyze interpolation and Beth definability for those fragments. In contrast with expressive power or decidability, where in many cases one can have some intuition about the expected results, interpolation and Beth definability are properties with a more elusive behavior. Speaking in a general way, extending or restricting a logic may cause to gain or lose these properties, and there is not a "general recipe" that can be applied uniformly. We want to analyze how memory logics behave from this perspective, and to see whether the standard techniques are fruitful in this context.

As we said in Section 2.2.3, when we work with hybrid logics there is a choice concerning the inclusion of nominals in the common language. In the context of memory logics, we should also decide if we want to include ⓚ or not. The first possibility is to keep the already defined notion of interpolation over propositional symbols, where the common language restricts only the occurrences of propositional symbols. The other option is to include ⓚ in the restrictions, and allow ⓚ to occur in the interpolant only when ⓚ is present in both the antecedent and the consequent. We will refer to this last option as *interpolation over propositional symbols and known*. It is clear that this option is stronger than interpolation over propositional symbols, since the interpolant must fulfill more requirements.

Naturally, in the context of hybrid memory logics we have to decide *both* things: whether we want to consider nominals in the common language and, independently, the ⓚ operator. We are not going to study every possible combination here. We will cover just some memory fragments, and show how they behave in terms of these properties.

5.1 Interpolation fails

We are going to show that most of the memory logics we are studying lack interpolation (with the exception of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ and $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$). We will use a classic technique to prove this, whose general schema is the following. First, we define φ and ψ such that $\varphi \to \psi$ is a valid formula. Then, we find two models $\langle \mathcal{M}, w\rangle$ and $\langle \mathcal{M}', w'\rangle$, such that w and w' are bisimilar in the common language of φ and ψ, but $\mathcal{M}, w \models \varphi$ and $\mathcal{M}', w' \models \neg\psi$. These conditions are enough to claim that interpolation cannot hold. Why? For suppose not, then there is an interpolant χ in the common language of φ and ψ such that $\varphi \to \chi \to \psi$ is valid. Therefore χ holds at $\langle \mathcal{M}, w\rangle$. Because w and w' are bisimilar in the common language, χ also holds at $\langle \mathcal{M}', w'\rangle$. This implies that ψ holds at $\langle \mathcal{M}', w'\rangle$ too, but this is an absurd, since we assumed that $\neg\psi$ holds there.

Some remarks before we begin presenting the results. When the context is clear enough we just say "the common language" referring to one of the possible definitions for this (i.e., over propositional symbols, over propositional symbols and known, etc.). Second, observe that $\top$ can always occur in the interpolant, since otherwise the definition of interpolation can be easily trivialized. Finally, unless we explicitly say otherwise, we prove interpolation (or the lack thereof) for the class of all models.

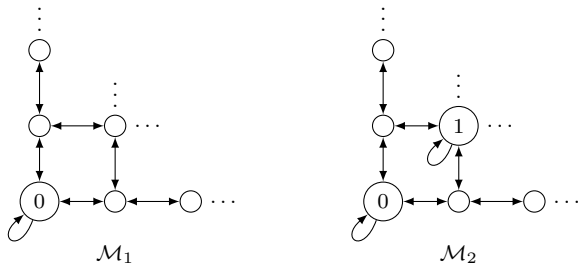
We begin by showing failure of interpolation for $\mathcal{ML}^m_\emptyset(\langle r\rangle)$, and how this can be transfered to other memory logics. Since we need to find two bisimilar models in the common language of φ and ψ, this result is strongly based on the bisimilar models we founded in Section 3.2. We are going to prove the failure of interpolation over propositional symbols. This implies, of course, the same failure over propositional symbols and known.

5.1.1. Theorem. *$\mathcal{ML}^m_\emptyset(\langle r\rangle)$ lacks interpolation over propositional symbols.*

Proof. Let $\varphi = q \wedge \textcircled{r}[r](\neg\textcircled{k} \to \varphi')$. If $\mathcal{M}, w \models \varphi$ then q is true at w and any successor of w different from w satisfies φ'. Now, let $\varphi' = \neg q \wedge \neg\textcircled{r}\langle r\rangle(\textcircled{k} \wedge \neg q)$. With this definition of φ', if $\mathcal{M}, w \models \varphi$ then for all v such that wRv and $v \neq w$ we have $\neg vRv$.

Let $\psi = p \wedge \langle r\rangle(\neg p \wedge \textcircled{r}\langle r\rangle\textcircled{k})$. If $\mathcal{M}, w \models \psi$ then there is $v \neq w$ such that wRv and vRv. It is clear that $\varphi \wedge \psi$ is a contradiction, so $\varphi \to \neg\psi$ is valid.

Let $\mathcal{M}_1 = \langle \omega, R_1, \emptyset, \emptyset\rangle$ and $\mathcal{M}_2 = \langle \omega, R_2, \emptyset, \emptyset\rangle$, where $R_1 = \{(n, m) \mid n \neq m\} \cup \{(0, 0)\}$ and $R_2 = R_1 \cup \{(1, 1)\}$. Graphically,



where the accessibility relation is the transitive closure of the arrows shown but without reflexive loops excepts those explicitly marked. In Theorem 3.2.4 it was shown that $\langle \mathcal{M}_1, 0\rangle$ and $\langle \mathcal{M}_2, 0\rangle$ are bisimilar over $\mathcal{ML}^m_\emptyset(\langle r\rangle)$. Now, define the models $\mathcal{M}'_1$ and $\mathcal{M}'_2$ as $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively but with a nonempty valuation in the following way: $\mathcal{M}'_1 = \langle \omega, R_1, V_1, \emptyset\rangle$ and $\mathcal{M}'_2 = \langle \omega, R_2, V_2, \emptyset\rangle$, where $R_1 = \{(n,m) \mid n \neq m\} \cup \{(0,0)\}$, $R_2 = R_1 \cup \{(1,1)\}$, $V_1(q) = \{0\}$ and $V_2(p) = \{0\}$. One can verify that $\mathcal{M}'_1, 0$ and $\mathcal{M}'_2, 0$ are bisimilar over the common language and that $\mathcal{M}'_1, 0 \models \varphi$ and $\mathcal{M}'_2, 0 \models \psi$.

Suppose there is an interpolant χ over the common language of φ and ψ for the valid formula $\varphi \to \neg\psi$. On the one hand, since φ is true at $\langle \mathcal{M}'_1, 0\rangle$ then χ also is. On the other, since ψ is true at $\langle \mathcal{M}'_2, 0\rangle$ then $\neg\chi$ also is. Then we have that $\mathcal{M}'_1, 0 \models \chi$ and $\mathcal{M}'_2, 0 \models \neg\chi$, which is an absurd because $\langle \mathcal{M}'_1, 0\rangle$ and $\langle \mathcal{M}'_2, 0\rangle$ are bisimilar over the common language. □

From this result there are a couple of corollaries that can be formulated. First, we can apply here the same idea we used in Lemma 4.2.5 (to force a "clean" zone of depth d from a given point) to transfer the above result to $\mathcal{ML}^m(\langle r\rangle)$.

5.1.2. Corollary. *$\mathcal{ML}^m(\langle r\rangle)$ lacks interpolation over propositional symbols.*

Proof. Let φ and ψ be as in Theorem 5.1.1. Let $\theta = \neg ⓚ \wedge [r]\neg ⓚ \wedge [r][r]\neg ⓚ$. Define $\varphi' = \varphi \wedge \theta$ and $\psi' = \psi \wedge \theta$ and repeat the argument of the above proof. □

The second corollary uses the idea of the translation from $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ to $\mathcal{ML}^m_\emptyset(\langle r\rangle)$ we defined in Theorem 3.2.2.

5.1.3. Corollary. *$\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ lacks interpolation over propositional symbols.*

Proof. Observe that in the proof of Theorem 5.1.1, instead of ψ, one could use $\psi' = p \wedge ⓡ\langle r\rangle(\neg p \wedge ⓡ\langle r\rangle(ⓚ \wedge \neg p))$ instead. Now, in both φ and ψ', all occurrences of $\langle r\rangle$ are of the form $ⓡ\langle r\rangle$, and all occurrences of $[r]$ are of the form $ⓡ[r]$. Therefore they can be translated to $\langle\langle r\rangle\rangle$ and $[\![r]\!]$ preserving equivalence. Since $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ is less expressive than $\mathcal{ML}^m_\emptyset(\langle r\rangle)$, both models of the proof of Theorem 5.1.1 are $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$-bisimilar and therefore the argument is valid. □

Now we are going to analyze interpolation when we add the operators ⓔ and ⓕ (and we use the classic diamond $\langle r\rangle$). The case of ⓔ is not hard to deal with, since we can reuse the models we already found in Section 2.2.1 to define a pair of bisimilar models in the common language. In contrast, we will leave the case of ⓕ open, since we could not find an equivalent pair of models for this case. The analysis of interpolation for this operator will probably have to wait until we find an answer to Question 3.2.16.

5.1.4. THEOREM. *$\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ)$ lacks interpolation over propositional symbols.*

PROOF. Let

$$\theta(q) = ⓡ\langle r\rangle(q \wedge ⓚ \wedge \neg\langle r\rangle(\neg q \wedge ⓚ)).$$

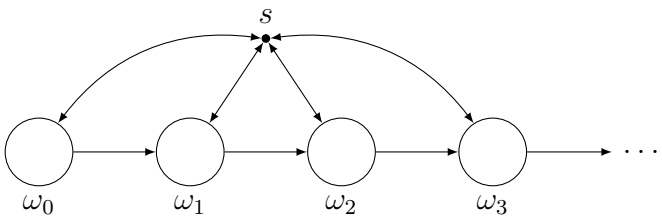
Suppose $\mathcal{M}$ is a model with $S = \{w\}$ where $\mathcal{M}, w \models q$ and $\mathcal{M}, v \models \neg q$. It is not difficult to see that $\mathcal{M}, v \models \theta(q)$ iff vRw and $\neg wRv$.

Now, let φ and ψ be the formulas

$$\begin{array}{rcl} \varphi & = & q \wedge ⓡ\langle r\rangle\langle r\rangle(\neg q \wedge \theta(q)) \\ \psi & = & p \wedge ⓡ[r][r](\neg ⓚ \to (\neg p \wedge \neg\theta(p))) \end{array}$$

(here $\theta(p)$ result of replacing all occurrences of q by p in the formula $\theta(q)$). On the one hand, if φ is true at a point w then there are points u and $v \neq w$ such that $wRuRv$ and vRw and $\neg wRv$. On the other hand, if ψ is true at a point w then for all points u and $v \neq w$ such that $wRuRv$ it is not the case that and vRw and $\neg wRv$. Hence $\models \varphi \to \neg\psi$.

Recall the model $\mathcal{M}$ we used in Theorem 3.2.13, where $\mathcal{M} = \langle\{s\} \cup \omega_0 \cup \omega_1 \cup \ldots, R, \emptyset, \emptyset\rangle$, where each ω_i is a different copy of ω, and $R = \{(n, m) \mid n \in \omega_i, m \in \omega_j, i \leq j\} \cup \{(n, s), (s, n) \mid \text{for all } n \neq s\}$. Graphically,



In Theorem 3.2.13 we showed that $\langle\mathcal{M}, w_0\rangle$ and $\langle\mathcal{M}, w_1\rangle$ are $\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ)$-bisimilar, where $w_0 \in \omega_0$ and $w_1 \in \omega_1$. Let $\mathcal{M}'$ be as $\mathcal{M}$ but with a nonempty valuation: $V(p) = \{w_0\}$, $V(q) = \{w_1\}$, and $V(r) = \emptyset$ for all $r \in$ PROP different from p and q. It is straightforward to verify that $\mathcal{M}', w_0 \models \psi$ and $\mathcal{M}', w_1 \models \varphi$, but $\langle\mathcal{M}', w_0\rangle$ and $\langle\mathcal{M}', w_1\rangle$ are $\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ)$-bisimilar in the common language. □

5.1.5. COROLLARY. *$\mathcal{ML}^m(\langle r\rangle, ⓔ)$ lacks interpolation over propositional symbols.*

PROOF. Let φ and ψ be as in Theorem 5.1.4. It is easy to see that $ⓔ\varphi \to \neg ⓔ\psi$ is a valid formula in the class of $\mathcal{ML}^m(\langle r\rangle, ⓔ)$-models. The rest of the argument is similar. □

5.2 Some positive results

In this section we are going to show that $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ has interpolation over propositional symbols and known with respect to a quite general frame class. The technique we use here is similar to the one presented in [tC05]. To develop the proof we will need some tools from the area of model theory so, before we go into the main theorem, we are going to give some definitions and preliminary results. For a more detailed treatment of these concepts, see [Doe96]. Once the result is established, using the translation we defined in Section 4.1.1 we will show that interpolation for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ easily follows.

Throughout this section, $\leftrightarrows$ will refer for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation. When we want to refer to an $\mathcal{ML}$-bisimulation, we will use $\leftrightarrows_{\mathcal{ML}}$.

5.2.1 Preliminaries

The first notion we are going to introduce is that of ω-saturatedness. This is a classic notion defined for first order models that we are going to apply to Kripke models, using the correspondence between Kripke and first order models we discussed in Section 2.2.1.

A 1-type is a set of first order formulas in one free variable. A 1-type $\Gamma(x)$ is *realized* in a first order model $\mathcal{M}$ if there is an element d of the domain of $\mathcal{M}$ such that $\mathcal{M}, g^x_d \models \Gamma$. A model is said to be 1-saturated if for all 1-types $\Gamma(x)$, if every finite subset of $\Gamma(x)$ is realized in $\mathcal{M}$, then $\Gamma(x)$ itself is realized in $\mathcal{M}$. One can think of 1-saturatedness as a sort of compactness within a model.

Given a model $\mathcal{M}$ and a finite sequence $d_1, \ldots, d_n$ of elements of the domain of $\mathcal{M}$, we use $(\mathcal{M}, d_1, \ldots, d_n)$ to denote the expansion of $\mathcal{M}$ in which the elements $d_1, \ldots, d_n$ are named by additional constants $c_1, \ldots, c_n$ (each new constant c_k denotes the corresponding element d_k in the expanded model).

5.2.1. DEFINITION. [ω-saturated model] A model $\mathcal{M}$ is ω-saturated if every such expansion $(\mathcal{M}, d_1, \ldots, d_n)$ (with $n \in \omega$) is 1-saturated.

As we said before, the concept of ω-saturatedness can be applied to Kripke models. In the same way, it can also be applied to memory Kripke models: to talk about a memory Kripke model $\langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ using first order logic we can use the standard first order correspondence language to interpret W, $(R_r)_{r\in\text{REL}}$ and V, plus a unary relation symbol *known* to interpret the memory S.

Now we are going to prove that ω-saturatedness is preserved under the operation of memorizing a finite set of elements.

5.2.2. Lemma. *Let $\mathcal{M}$ be an ω-saturated memory Kripke model. For any finite $A \subseteq M$ $\mathcal{M}[A]$ is ω-saturated.*

Proof. We will show that if every finite subset of $\Gamma(x)$ is realized in $\mathcal{M}[A]$ then $\Gamma(x)$ is realized in $\mathcal{M}[A]$.

Let $A = \{a_1, \ldots, a_n\}$ and let $\mathcal{M}(a_1, \ldots, a_n)$ denote the expansion of $\mathcal{M}$ in which the elements $a_1, \ldots, a_n$ are named by additional constants $c_1, \ldots, c_n$ (each new constant c_k denotes the corresponding element a_k in the expanded model). Since $\mathcal{M}$ is ω-saturated then $\mathcal{M}(a_1, \ldots, a_n)$ is 1-saturated.

We formulate the following easy claim, without a proof.

5.2.3. Claim. Let $\Gamma(x)$ be a 1-type set of formulas. Let $\tilde{\Gamma}(x) = \{\varphi[known(y)/known(y) \vee y = c_1 \vee \cdots \vee y = c_n] \mid \varphi \in \Gamma(x)\}$. Then $\mathcal{M}[A], g^x_d \models \Gamma$ iff $\mathcal{M}(a_1, \ldots, a_n), g^x_d \models \tilde{\Gamma}$.

Let $\Gamma(x)$ be a 1-type set of formulas such that every finite $\Gamma'(x) \subseteq \Gamma(x)$ is realized in $\mathcal{M}[A]$. By Claim 5.2.3, every finite subset of $\tilde{\Gamma}$ is realized in $\mathcal{M}(a_1, \ldots, a_n)$ and, since $\mathcal{M}(a_1, \ldots, a_n)$ is 1-saturated, $\tilde{\Gamma}$ also is. Therefore again by Claim 5.2.3, we conclude that Γ is realized in $\mathcal{M}[A]$. Since $\mathcal{M}$ is ω-saturated, it is trivial to see that $\mathcal{M}[A]$ also is. □

Why are we interested in ω-saturated models? We already saw that if two points are bisimilar, then they are modally equivalent. The converse does not hold in general, but ω-saturated models will allow us to turn modal equivalence for $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ into a bisimulation. The problem is that we are going to start working with models that are not necessarily ω-saturated, so we will need to have equivalent ω-saturated models. There is a classic theorem that guarantees that this is always possible.

5.2.4. Definition. [Elementary extension] A first order model $\mathcal{N}$ is an *extension* of a model $\mathcal{M}$ if $\mathcal{M}$ is a subset of the domain of $\mathcal{N}$ and the interpretation of every non-logical symbol in $\mathcal{M}$ is simply the restriction of its interpretation in $\mathcal{N}$ with respect to the domain of $\mathcal{M}$. We say that $\mathcal{N}$ is an *elementary* extension of $\mathcal{M}$ if it is an extension and for all first order formulas $\varphi(x_1, \ldots, x_n)$ and elements $d_1, \ldots, d_n$ of the domain of $\mathcal{M}$, $\mathcal{M}, g \models \varphi$ iff $\mathcal{N}, g' \models \varphi$, where $g(x_i) = g'(x_i) = d_i$ for all $1 \leq i \leq n$.

5.2.5. Theorem ([Doe96]). *Every model $\mathcal{M}$ has an ω-saturated elementary extension $\mathcal{M}^+$.*

The next notion we are going to introduce is that of bisimulation products.

5.2.6. Definition. A bisimulation product of a set of frames $\{\mathcal{F}_i \mid i \in I\}$ is a subframe $\mathcal{B}$ of the Cartesian product $\Pi_i \mathcal{F}_i$ such that for each $i \in I$, the natural projection function $f_i : \mathcal{B} \to \mathcal{F}_i$ is a surjective bounded morphism.

This operation, together with the following theorem, enable us to construct a new frame using a total bisimulation between two given frames. This will be helpful later to construct a model that will act as a witness for the interpolant.

5.2.7. THEOREM ([MV97]). *Let $\mathcal{H}$ be a submodel of the product $\mathcal{F} \times \mathcal{G}$. Then $\mathcal{H}$ is a bisimulation product of $\mathcal{F}$ and $\mathcal{G}$ iff the domain of $\mathcal{H}$ is a total frame bisimulation between $\mathcal{F}$ and $\mathcal{G}$.*

Finally, we introduce the concept of total bisimulation in the context of memory logics. The intuitive notion is that it is a bisimulation in which every possible pairs are related.

5.2.8. DEFINITION. [Total $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation] Let $\mathcal{M}$ and $\mathcal{N}$ be two models of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. We say that $\mathcal{M}, w$ and $\mathcal{N}, v$ are *totally bisimilar* ($\mathcal{M}, w \leftrightarrows^T \mathcal{N}, v$) when there is bisimulation $\sim$ between $\mathcal{M}, w$ and $\mathcal{N}, v$ and

1. for every $A = \{a_1, \ldots, a_k\} \subseteq M$ with $a_i R a_{i+1}$, and every $a \in M$ there is a $B = \{b_1, \ldots, b_k\} \subseteq N$ with $b_i R b_{i+1}$ and $b \in N$ such that $(A, a) \sim (B, b)$

2. for every $B = \{b_1, \ldots, b_k\} \subseteq N$, and every $b \in N$ there is $A = \{a_1, \ldots, a_k\} \subseteq M$ with $a_i R a_{i+1}$ and $a \in M$ such that $(A, a) \sim (B, b)$.

5.2.2 The main result

We first show a sketch of the proof. We start by supposing there are two $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas φ and ψ such that $\varphi \to \psi$ is valid, but it doesn't have an interpolant in the common language. In general, the bisimulations we discuss here between a pair of models are always established with respect to the common language of φ and ψ. We first show that there are two models $\mathcal{M}$ and $\mathcal{N}$ such that $\mathcal{M}, w \models \varphi$ and $\mathcal{N}, v \models \neg\psi$. We next take ω-saturated models $\mathcal{M}^+$ and $\mathcal{N}^+$ of $\mathcal{M}$ and $\mathcal{N}$ respectively and show $\mathcal{M}^+, w \leftrightarrows^T \mathcal{N}^+, v$. According to Proposition 4.1.3, we take equivalent tree $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-models $\mathcal{M}^+_T$ and $\mathcal{N}^+_T$ such that $\mathcal{M}^+, w \leftrightarrows^T \mathcal{M}^+_T, w$ and $\mathcal{N}^+, v \leftrightarrows^T \mathcal{N}^+_T, v$. We conclude $\mathcal{M}^+_T, w \leftrightarrows^T \mathcal{N}^+_T, v$.

Then we switch to the basic model logic $\mathcal{ML}$. Let $\mathcal{M}^+_{T_{\mathcal{ML}}}$ and $\mathcal{N}^+_{T_{\mathcal{ML}}}$ be the corresponding $\mathcal{ML}$-models of $\mathcal{M}^+_T$ and $\mathcal{N}^+_T$ respectively (shifting the signature to $\langle \text{PROP} \cup \{known\}, \text{REL}\rangle$). Being $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ an extension of $\mathcal{ML}$, we have $\mathcal{M}^+_{T_{\mathcal{ML}}} \leftrightarrows^T_{\mathcal{ML}} \mathcal{N}^+_{T_{\mathcal{ML}}}$. Using Theorem 5.2.7, one can show that there is a bisimulation product $\mathcal{H} \in \mathcal{C}$ of the frames of $\mathcal{M}'$ and $\mathcal{N}'$, and a valuation V such that $(\mathcal{H}, V), \langle w, v\rangle \models (\varphi \wedge \neg\psi)[ⓚ/known]$.

Since by its definition, $\mathcal{H}$ is tree like, we can return to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ and conclude that $\varphi \wedge \neg\psi$ is satisfiable in some $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-model of $\mathcal{C}$, contradicting our hypothesis. Graphically, the general schema is the following (the double headed arrows represents total bisimulations):

$$
\begin{tikzcd}
\mathcal{M} \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & \mathcal{N} \arrow[d, leftrightarrow] \\
\mathcal{M}^+ \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & \mathcal{N}^+ \arrow[d, leftrightarrow] \\
\mathcal{M}^+_T \arrow[r, leftrightarrow] \arrow[d, equal] & \mathcal{N}^+_T \arrow[d, equal] \\
\mathcal{M}^+_{T_{\mathcal{ML}}} \arrow[r, leftrightarrow, "\mathcal{ML}"] & \mathcal{N}^+_{T_{\mathcal{ML}}} \\
& (\mathcal{H}, V) \arrow[ul, bend left, "\mathcal{ML}"] \arrow[ur, bend right, "\mathcal{ML}"']
\end{tikzcd}
$$

5.2.9. Theorem. *Let $\mathcal{C}$ be any elementary frame class closed under generated subframes and bisimulation products. Then $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ has interpolation over propositions and known relative to $\mathcal{C}$.*

Proof. Let φ and ψ such that $\mathcal{C} \models \varphi \to \psi$ and let $\mathcal{L}$ be the common language of φ and ψ. Suppose for the sake of contradiction that there is no interpolant of φ and ψ in the language $\mathcal{L}$. We first prove two easy lemmas.

5.2.10. Lemma. *There is a model $\mathcal{M}$ based on a frame in $\mathcal{C}$, with a world w, such that $\mathcal{M}, w \models \{\chi \mid \mathcal{C} \models \varphi \to \chi$ and $\chi \in \mathcal{L}\} \cup \{\neg\psi\}$.*

Proof. By compactness, it suffices to show that every finite subset of $\mathsf{cons}(\varphi) \cup \{\neg\psi\}$ is satisfiable on $\mathcal{C}$. Consider any $\chi_1, \ldots, \chi_n \in \mathsf{cons}(\varphi)$. If $\{\chi_1, \ldots, \chi_n, \neg\psi\}$ wouldn't be satisfiable on $\mathcal{C}$, then $\chi_1 \wedge \cdots \wedge \chi_n$ would be an interpolant for $\varphi \to \psi$. By assumption, $\varphi \to \psi$ has no interpolant, and therefore, $\{\chi_1, \ldots, \chi_n, \neg\psi\}$ is satisfiable on $\mathcal{C}$. $\square$

Since $\mathcal{C}$ is closed under generated subframes we may assume that $\mathcal{M}$ is generated by w.

5.2.11. Lemma. *There is a model $\mathcal{N}$ based on a frame in $\mathcal{C}$, with a world v, such that $\mathcal{N}, v \models \{\chi \mid \mathcal{M}, w \models \chi$ and $\chi \in \mathcal{L}\} \cup \{\varphi\}$.*

Proof. Recall that $T(\mathcal{M}, w)$ denotes the set of formulas satisfied at $\langle \mathcal{M}, w \rangle$. By compactness, it suffices to show that every finite subset of $T(\mathcal{M}, w) \cup \{\varphi\}$ is satisfiable on $\mathcal{C}$. Consider any $\chi_1, \ldots, \chi_n \in T(\mathcal{M}, w)$. Suppose for the sake of contradiction that $\{\chi_1, \ldots, \chi_n, \varphi\}$ is not satisfiable on $\mathcal{C}$. Then $\mathcal{C} \models \varphi \to \neg(\chi_1 \wedge \cdots \wedge \chi_n)$. Hence, $\neg(\chi_1 \wedge \cdots \wedge \chi_n) \in \mathsf{cons}(\varphi)$, and therefore, $\mathcal{M}, w \models \neg(\chi_1 \wedge \cdots \wedge \chi_n)$. This contradicts the fact that $\chi_1, \ldots, \chi_n \in T(\mathcal{M}, w)$. $\square$

Again we may assume that $\mathcal{N}$ is generated by v. Let $\mathcal{M}^+$ and $\mathcal{N}^+$ be ω-saturated elementary extensions of $\mathcal{M}$ and $\mathcal{N}$ respectively. Let's suppose that the first order models $\mathcal{M}^+$ and $\mathcal{N}^+$ have domains M and N and binary relations R_1 and R_2 for the modal operator $\langle r\rangle$, respectively.

We define the relation $\sim$ between $\wp(M)\times M$ and $\wp(N)\times N$ in the following way: for all finite $A\subseteq M$ and finite $B\subseteq N$,

$$(A,a)\sim(B,b) \text{ iff for all formulas } \chi \text{ in } \mathcal{L},\ \mathcal{M}^+[A],a\models\chi \text{ iff } \mathcal{N}^+[B],b\models\chi.$$

Notice that by construction $(\emptyset,w)\sim(\emptyset,v)$. We prove now that $\sim$ is indeed a bisimulation. One remark first, in the proof below, ST_x is the standard translation from $\mathcal{ML}^m(\langle r\rangle)$ formulas to first order logic formulas we discussed in Section 3.3.

5.2.12. Lemma. *$\sim$ is an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation between $\mathcal{M}^+$ and $\mathcal{N}^+$ with respect to $\mathcal{L}$.*

Proof. By the definition of $\sim$, it is clear that the condition (*agree*) of Definition 3.1.1 is satisfied, restricted to $\mathcal{L}$. Let us see (*mzig*). Suppose $(A,a)\sim(B,b)$ and aR_1a'. Let

$$\Gamma=\{\mathsf{ST}_x(\chi)\mid \mathcal{M}^+[A\cup\{a\}],a'\models\chi \text{ and } \chi\in\mathcal{L}\}.$$

Let c_b be a new constant denoting the element b of $\mathcal{N}^+$. We next show that $\Gamma\cup\{R(c_b,x)\}$ is realized in $\mathcal{N}^+[B\cup\{b\}]$, where R is the first order binary relation symbol for $\langle\langle r\rangle\rangle$. Since, by Lemma 5.2.2, the expansion of $\mathcal{N}^+[B\cup\{b\}]$ with the constant c_b is 1-saturated, it suffices to show that every finite subset of Γ is realized in $\mathcal{N}^+[B\cup\{b\}]$ by an R_2-successor of b. Let $\mathsf{ST}_x(\chi_1),\dots,\mathsf{ST}_x(\chi_n)\in\Gamma$. We have $\mathcal{M}^+[A],a\models\langle\langle r\rangle\rangle(\chi_1\wedge\cdots\wedge\chi_n)$, and therefore $\mathcal{N}^+[B],b\models\langle\langle r\rangle\rangle(\chi_1\wedge\cdots\wedge\chi_n)$, which implies that there is an R_2-successor of b which satisfies $\chi_1\wedge\cdots\wedge\chi_n$. I.e., in $\mathcal{N}^+[B\cup\{b\}]$ there is an R_2-successor which realizes $\{\mathsf{ST}_x(\chi_1),\dots,\mathsf{ST}_x(\chi_n)\}$. Hence, there is b', bR_2b' such that $\mathcal{N}^+[B\cup\{b\}],g^x_{b'}\models\Gamma$. Therefore for every χ of $\mathcal{L}$, if $\mathcal{M}^+[A\cup\{a\}],a'\models\chi$ then $\mathcal{N}^+[B\cup\{b\}],b'\models\chi$. To see the other implication, suppose by contradiction that $\mathcal{N}^+[B\cup\{b\}],b'\models\chi$ but $\mathcal{M}^+[A\cup\{a\}],a'\not\models\chi$ (the case $\mathcal{M}^+[A\cup\{a\}],a'\models\chi$ but $\mathcal{N}^+[B\cup\{b\}],b'\not\models\chi$ is similar). This would imply that $\mathcal{M}^+[A\cup\{a\}],a'\models\neg\chi$ and hence $\mathcal{N}^+[B\cup\{b\}],b'\models\neg\chi$ which is an absurd. The (*mzag*) condition is similar.

To see (*remember*), suppose that $\mathcal{M}^+[A],a\models\chi$ iff $\mathcal{N}^+[B],b\models\chi$ for all χ of $\mathcal{L}$. Now, let χ be any formula of $\mathcal{L}$. By hypothesis, $\mathcal{M}^+[A],a\models \textcircled{\mathfrak{r}}\chi$ iff $\mathcal{N}^+[B],b\models\textcircled{\mathfrak{r}}\chi$. Applying the definition of $\textcircled{\mathfrak{r}}$, we obtain $\mathcal{M}^+[A\cup\{a\}],a\models\chi$ iff $\mathcal{N}^+[B\cup\{b\}],b\models\chi$. □

The following lemma helps prove that the bisimulation $\sim$ we have defined is total.

5.2.13. Lemma. *For every $a \in M$ there is $b \in N$ such that $(\emptyset, a) \sim (\emptyset, b)$; also for every $b \in N$ there is $a \in M$ such that $(\emptyset, a) \sim (\emptyset, b)$*

Proof. We prove the first assertion, since the second es similar. Let $a \in M$ and let

$$\Gamma = \{\mathsf{ST}_x(\chi) \mid \mathcal{M}^+, a \models \chi \text{ and } \chi \in \mathcal{L}\}$$

We need to show that Γ is realized in $\mathcal{N}^+$. By ω-saturatedness, it suffices to show that every finite subset of Γ is realized in $\mathcal{N}^+$. Let $\mathsf{ST}_x(\chi_1), \ldots, \mathsf{ST}_x(\chi_n) \in \Gamma$. Then, if $\theta = \mathsf{ST}_x(\chi_1) \wedge \cdots \wedge \mathsf{ST}_x(\chi_n)$, the formula $(\exists x)\theta$ is true at $\mathcal{M}^+$ and therefore also at $\mathcal{M}$ (recall that $\mathcal{M}^+$ is an elementary extension of $\mathcal{M}$). Since $\mathcal{M}$ is generated by w, there is $n \geq 0$ such that $\mathcal{M}, w \models \langle\langle r \rangle\rangle^{(n)}\theta$. Since $(\emptyset, w) \sim (\emptyset, v)$, we have that $\mathcal{N}, v \models \langle\langle r \rangle\rangle^{(n)}\theta$. Since $\mathcal{N}^+$ is an elementary extension of $\mathcal{N}$ it follows that $\mathcal{N}^+, v \models \langle\langle r \rangle\rangle^{(n)}\theta$. We conclude that there is $b \in N$ such that $\mathcal{N}^+, b \models \theta$, and so $\{\mathsf{ST}_x(\chi_1), \ldots, \mathsf{ST}_x(\chi_n)\}$ is realized in $\mathcal{N}^+$. □

5.2.14. Corollary. *The $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$-bisimulation $\sim$ is total.*

Proof. By a straightforward induction, using *(mforth)* and *(mback)* conditions of $\sim$. □

Applying the tree model property for $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ (see Theorem 4.1.3), let $\mathcal{M}^+_T$ and $\mathcal{N}^+_T$ be tree $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$-models such that $\mathcal{M}^+, w \leftrightarrows^T \mathcal{M}^+_T, w$ and $\mathcal{N}^+, v \leftrightarrows^T \mathcal{N}^+_T, v$. By Corollary 5.2.14, $\mathcal{M}^+, w \leftrightarrows^T \mathcal{N}^+, v$, and by transitivity of total bisimulations, we conclude $\mathcal{M}^+_T, w \leftrightarrows^T \mathcal{N}^+_T, v$.

Now, let $\mathcal{M}^+_{T_{\mathcal{ML}}}$ and $\mathcal{N}^+_{T_{\mathcal{ML}}}$ be the $\mathcal{ML}$ equivalent models for $\mathcal{M}^+_T$ and $\mathcal{N}^+_T$. Since $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$-bisimulation implies $\mathcal{ML}$-bisimulation, $\mathcal{M}^+_{T_{\mathcal{ML}}} \leftrightarrows^T_{\mathcal{ML}} \mathcal{N}^+_{T_{\mathcal{ML}}}$. Let $\mathcal{F}$ and $\mathcal{G}$ be the underlying frames of $\mathcal{M}^+_{T_{\mathcal{ML}}}$ and $\mathcal{N}^+_{T_{\mathcal{ML}}}$ respectively. Using Theorem 5.2.7, we know there is a bisimulation product $\mathcal{H} \in \mathcal{C}$ of $\mathcal{F}$ and $\mathcal{G}$ of which the domain is $\sim$. By the definition of bisimulation products, the natural projections $f : \mathcal{H} \to \mathcal{F}$ and $g : \mathcal{H} \to \mathcal{G}$ are surjective bounded morphisms. For any proposition letter $p \in prop(\varphi)$, let $V(p) = \{u \mid \mathcal{M}_{T_{\mathcal{ML}}}, f(u) \models p\}$, and for any proposition letter $p \in \mathsf{props}(\psi)$, let $V(p) = \{u \mid \mathcal{N}^+_{T_{\mathcal{ML}}}, g(u) \models p\}$. The properties of $\sim$ guarantee that this V is well-defined for $p \in \mathsf{props}(\varphi) \cap \mathsf{props}(\psi)$. By a standard argument, the graph of f is a bisimulation between $(\mathcal{H}, V)$ and $\mathcal{M}^+_{T_{\mathcal{ML}}}$ with respect to $\mathsf{props}(\varphi)$, and the graph of g is a bisimulation between $(\mathcal{H}, V)$ and $\mathcal{N}^+_{T_{\mathcal{ML}}}$ with respect to $\mathsf{props}(\psi)$.

Now we have the appropriate model in which the contradiction is made explicit, but we have to be able to raise this result to $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$. Notice that the model $(\mathcal{H}, V)$ is a tree, since it is the bisimulation product of two trees, and also that the signature of $(\mathcal{H}, V)$ is $\langle \textsc{prop} \cup \{known\}, \textsc{rel}\rangle$. Therefore, we can define the $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$-model $(\mathcal{H}, V', S)$ over $\langle \textsc{prop}, \textsc{rel}\rangle$ where $V' = V$ for all $p \in \textsc{prop}$ and $w \in V(known)$ iff $w \in S$. It is easy to see that the equivalent $\mathcal{ML}$-model for $(\mathcal{H}, V', S)$ is $(\mathcal{H}, V)$. So now we need some claim that guarantee

us that we can build two relations $\sim_f$ and $\sim_g$ from the graphs of f and g respectively, such that $\sim_f$ is an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation between $(\mathcal{H}, V', S)$ and $\mathcal{M}_T^+$ and $\sim_g$ is an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation between $(\mathcal{H}, V', S)$ and $\mathcal{N}_T^+$. We will leave this as a lemma to be proved later. Assuming that we can actually build those relations, it follows that $(\mathcal{H}, V', S), \langle w, v\rangle \models \varphi \wedge \neg\psi$. This contradicts our initial assumption that $\mathcal{C} \models \varphi \to \psi$. □

We only left to prove the following lemma.

5.2.15. Lemma. *Let $\mathcal{M}$ and $\mathcal{N}$ be two tree $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-models over $\langle$PROP, REL$\rangle$ such that $\mathcal{M}'$ and $\mathcal{N}'$ are their corresponding tree $\mathcal{ML}$-models over the signature $\langle$PROP $\cup \{known\}$, REL$\rangle$ (recall the association we defined in Section 3.2.4). Let $\sim_{\mathcal{ML}}$ be an $\mathcal{ML}$-bisimulation between $\langle\mathcal{M}', w\rangle$ and $\langle\mathcal{N}', v\rangle$ where w and v are their corresponding tree roots. Let $Z \subseteq (\wp(M) \times M) \times (\wp(N) \times N)$ be the relation $(A, w_n)\ Z\ (B, v_n)$ iff $w_n \sim_{\mathcal{ML}} v_n$, and either $A = \{w_0, \ldots, w_{n-1}\}$ and $B = \{v_0, \ldots, v_{n-1}\}$, or $A = \{w_0, \ldots, w_{n-1}, w_n\}$ and $B = \{v_0, \ldots, v_{n-1}, v_n\}$, where $w_0 = w$, $v_0 = v$ and $w_i R w_{i+1}$ and $v_i R v_{i+1}$ for $i \in \{0, \ldots, n-1\}$. Then Z is an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation between $\langle\mathcal{M}, w\rangle$ and $\langle\mathcal{N}, v\rangle$.*

Proof. We should check that Z is an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation between $\mathcal{M}, w$ and $\mathcal{N}, v$. The relation Z satisfies *non-triv* since $w \sim_{\mathcal{ML}} v$, and therefore by definition $(\emptyset, w) Z (\emptyset, v)$. By definition of Z, we have only to cases where $(A, w) Z (B, v)$, so let us suppose that (1) $((w_0, \ldots, w_{n-1}), w_n) Z ((v_0, \ldots, v_{n-1}), v_n)$, or (2) $((w_0, \ldots, w_n), w_n)\ Z\ ((v_0, \ldots, v_n), v_n)$, and let check the rest of the conditions. Since $w_n \sim_{\mathcal{ML}} v_n$, we know that w_n and v_n coincide in the propositional symbols, and that includes the proposition *known*. So, to check that agree holds we only have to see what happens with ⓚ. If we are in case (1), by the correspondence between a $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ model and an $\mathcal{ML}$ model, we known that $\mathcal{M}, w \models_{\mathcal{ML}^m(\langle\langle r\rangle\rangle)}$ ⓚ iff $\mathcal{M}', w \models_{\mathcal{ML}}$ *known* (and the same for $\mathcal{N}$ and v). Since we have tree-like models, we know that $w_0, \ldots, w_{n-1}, w_n$ are all different points, and therefore $\mathcal{M}[w_0, \ldots, w_{n-1}], w_n \models$ ⓚ iff $\mathcal{M}, w_n \models$ ⓚ. Therefore, w_n and v_n coincides in the propositional symbols and known, and *agree* is satisfied. If we are in case (2), ⓚ holds both in $\mathcal{M}[w_0, \ldots, w_n], w_n$ and $\mathcal{N}[v_0, \ldots, v_n], v_n$, and so *agree* is also satisfied. To see *mforth*, let suppose that $R_{\mathcal{M}}(w_n, w_{n+1})$. Since $w_n \sim_{\mathcal{ML}} v_n$, we know that there is a v_{n+1} such that $R_{\mathcal{N}}(v_n, v_{n+1})$ and $w_{n+1} \sim_{\mathcal{ML}} v_{n+1}$. By definition,

$$\begin{array}{rcl} ((w_0, \ldots, w_n), w_{n+1}) & Z & ((v_0, \ldots, v_n), v_{n+1}) \quad \text{and} \\ ((w_0, \ldots, w_n, w_{n+1}), w_{n+1}) & Z & ((v_0, \ldots, v_n, v_{n+1}), v_{n+1}), \end{array}$$

and that satisfies *mforth* for cases (1) and (2). The case for *mback* is analogous. The *remember* condition holds trivially by definition of Z. □

From this result, interpolation for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ is straightforward using the equivalence preserving translations we defined in Theorem 4.1.5 between $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ and $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$.

5.2.16. THEOREM. *Let $\mathcal{C}$ be any elementary frame class closed under generated subframes and bisimulation products. Then $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ has interpolation over propositions and known relative to $\mathcal{C}$.*

PROOF. Let Tr be the equivalence preserving translation defined in Theorem 4.1.5 that takes $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$-formulas to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas. Observe that Tr preserves propositional symbols and known, that is, given $\varphi \in \mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$, ⓚ occurs in φ if and only if ⓚ occurs in $\mathsf{Tr}(\varphi)$ and $\mathsf{props}(\varphi) = \mathsf{props}(\mathsf{Tr}(\varphi))$.

Let φ and ψ be two $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$-formulas such that $\varphi \to \psi$ is valid. Using Tr, we know that $\mathsf{Tr}(\varphi) \to \mathsf{Tr}(\psi)$ is a valid $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formula. By Theorem 5.2.9, we know that there is an interpolant χ for $\mathsf{Tr}(\varphi)$ and $\mathsf{Tr}(\psi)$ in the common language. Since Tr preserves equivalence, χ is also an interpolant for φ and ψ. Furthermore, given that Tr preserves propositional symbols and known, χ is in the common language of φ and ψ. □

5.3 The quest for Beth definability

In this section we want to analyze the Beth property for some of the memory fragments we are working with. As we said in Section 2.2.3, the Beth property is present usually when interpolation also is (see [Hoo01] for examples). Therefore, the expected result would be to have the Beth property for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, while for other memory fragments we studied, it should not be present.

Having proved interpolation for a given logic, the usual technique to prove Beth is based on "simulating" global satisfiability using local evaluations. This technique works for logics that have the generated submodel property. Let's give an intuitive idea of how to do this. Suppose that we are interested in models that globally satisfy a given set of formulas Σ, but we want to describe this condition by evaluating a formula locally, at a given point. How can we accomplish this? The idea is to take each formula of Σ and stack in front of it enough boxes to reach every possible point. More formally, we can construct a new set $\Gamma = \{\Box^n\varphi \mid \varphi \in \Sigma, n \in \omega\}$, and using the generated submodel property, take a submodel $\mathcal{M}_w$ of $\mathcal{M}$ generated from w, and evaluate Γ at w. Observe that there are always formulas in Γ with enough stacked boxes to reach from w every point in $\mathcal{M}_w$. Thus, if Γ is satisfied at w, Σ must hold globally. In this way, we can transform global semantic consequence in local semantic consequence, where interpolation can be applied to construct an explicit definition from an implicit one. Several examples of this type of proof can be found in [tC05].

The problem we face when we try to apply this technique for the case of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ is that we do not have the usual $[r]$ operator, but we have $[\![r]\!]$ instead. Recall that the operator $[\![r]\!]$ remembers the current point before making a modal transition, and therefore it is not clear that Γ works now to ensure global satisfiability. Why? The reason is that in this context, when a certain point is

reached the model may have several memorized points due to $[\![r]\!]$, and the original set might not hold there.

Nevertheless, the interaction between global satisfiability and the memory operators seems to leave a gap where this technique can be still applied. We could not find a proof of the Beth property for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, but we think that this property indeed holds. The proof for Beth depends on the following conjecture:

5.3.1. Conjecture. *Let Σ be a set of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas. If there is a model $\langle\mathcal{M}, w\rangle$ such that $\mathcal{M} \models \Sigma$ and there is not an acyclic model $\langle\mathcal{M}', w\rangle$ such that $\mathcal{M}, w \leftrightarrows \mathcal{M}', w$ and $\mathcal{M}' \models \Sigma$, then for all $\mathcal{M} = \langle W, (R_r)_{r\in\mathrm{REL}}, V, S\rangle$ such that $\mathcal{M} \models \Sigma$, $S = \emptyset$.*

Intuitively, this property says that if a set of formulas Σ imposes a cycle to all the models that satisfy Σ globally, then Σ also constraints the memory of the models to be empty.

We now show how Beth can be proved, assuming that the conjecture is correct. We give these results for the unimodal case, so we use $\langle\Diamond\rangle$ and $[\Box]$ instead of $\langle\langle r\rangle\rangle$ and $[\![r]\!]$.

5.3.2. Lemma. *Let Γ be a set of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas such that $\Gamma(p) \cup \Gamma(p') \models p \leftrightarrow p'$. Then there is a finite subset $\Gamma_0 \subseteq \Gamma$ and a formula δ in which p does not occur such that $\Gamma_0(p) \models p \leftrightarrow \delta$.*

Proof. By compactness, we know that there is a subset $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0(p) \cup \Gamma_0(p') \models p \leftrightarrow p'$. It follows that $\models (p \wedge \bigwedge \Gamma_0(p)) \to (\bigwedge \Gamma_0(p') \to p')$. By Theorem 5.2.9, there is an interpolant δ for the implication such that p and p' do not occur in δ. Hence, on one hand we have $\models (p \wedge \bigwedge \Gamma_0(p)) \to \delta$. On the other hand, $\models \delta \to (\bigwedge \Gamma_0(p') \to p')$, and by uniform substitution, $\models \delta \to (\bigwedge \Gamma_0(p) \to p)$. Therefore, we conclude $\models \delta \leftrightarrow (\bigwedge \Gamma_0(p) \to p)$. This implies $\Gamma_0(p) \models \delta \leftrightarrow p$ as desired. $\Box$

5.3.3. Theorem. *If Conjecture 5.3.1 is correct, then $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ has the Beth property with respect to the class of all models.*

Proof. Let Σ be a set of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas such that Σ implicitly defines the propositional letter p. Then, by definition of implicit definability, $\Sigma(p) \cup \Sigma(p') \models^{glo} p \leftrightarrow p'$. The first case is that for all $\mathcal{M}$ such that $\mathcal{M} \models \Sigma$, and for all w in $\mathcal{M}$, there is an acyclic model $\langle\mathcal{M}', w\rangle$ such that $\mathcal{M}, w \leftrightarrows \mathcal{M}', w$ and $\mathcal{M}' \models \Sigma$. Let $\Gamma = \{[\Box]^n\varphi \mid \varphi \in \Sigma, n \in \omega\}$. The following claim can be proved:

5.3.4. Claim. $\Gamma(p) \cup \Gamma(p') \models p \leftrightarrow p'$.

Proof. Suppose $\mathcal{M}, w \models \Gamma(p) \cup \Gamma(p')$ for some model $\mathcal{M}$. By Theorem 4.1.3, let $\mathcal{M}_T$ be a tree-like model such that $\mathcal{M}_T, w \leftrightarrows \mathcal{M}, w$. Since $\mathcal{M}_T$ is tree-like, we know that $\mathcal{M}_T[v], v \models \langle\Diamond\rangle\varphi$ iff $\mathcal{M}_T, v \models \langle\Diamond\rangle\varphi$ for all φ and v in $\mathcal{M}_T$. Therefore, by construction of Γ, $\mathcal{M}_T$ globally satisfies $\Sigma(p)$ and $\Sigma(p')$. Then it follows that $\mathcal{M}_T \models p \leftrightarrow p'$. Hence, $\mathcal{M}_T, w \models p \leftrightarrow p'$, and therefore $\mathcal{M}, w \models p \leftrightarrow p'$. □

By Lemma 5.3.2, there is a formula δ in which p does not occur such that $\Gamma_0(p) \models p \leftrightarrow \delta$. Let's see now that $\Sigma(p) \models^{glo} p \leftrightarrow \delta$. Suppose there is a model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma(p)$. Then for all w in $\mathcal{M}$, there is an acyclic model $\langle\mathcal{M}', w\rangle$ such that $\mathcal{M}, w \leftrightarrows \mathcal{M}', w$ and $\mathcal{M}' \models \Sigma$. Using again the fact that $\mathcal{M}'$ is acyclic, by construction of Γ we know that $\mathcal{M}', w \models \Gamma_0$. Therefore $\mathcal{M}', w \models p \leftrightarrow \delta$ and we conclude that $\mathcal{M}, w \models p \leftrightarrow \delta$. Therefore $\mathcal{M} \models p \leftrightarrow \delta$.

Let analyze now the other case. By Conjecture 5.3.1, we know that for all model $\mathcal{M}$, if $\mathcal{M} \models \Sigma$, then $\mathcal{M}$ has an empty memory. Let $\Gamma = \{[\Box]^n(\neg ⓚ \to \varphi) \mid \varphi \in \Sigma, n \in \omega\}$. We can prove a similar lemma as we did for the previous case:

5.3.5. Claim. $\Gamma(p) \cup \Gamma(p') \models p \leftrightarrow p'$.

Proof. Suppose $\mathcal{M}, w \models \Gamma(p) \cup \Gamma(p')$ for some model $\mathcal{M}$. Let $\mathcal{M}_w$ be the submodel of $\mathcal{M}$ generated by w. Since the memory of $\mathcal{M}$ is empty, by construction of Γ, $\mathcal{M}_w$ globally satisfies $\Sigma(p)$ and $\Sigma(p')$. Then it follows that $\mathcal{M}_w \models p \leftrightarrow p'$. Hence, $\mathcal{M}_w, w \models p \leftrightarrow p'$, and therefore $\mathcal{M}, w \models p \leftrightarrow p'$. □

Thus, in this case we can apply again Lemma 5.3.2 and conclude that there is a formula δ in which p does not occur and $\Gamma_0(p) \models p \leftrightarrow \delta$. Therefore, $\Sigma(p) \models^{glo} p \leftrightarrow \delta$. □

5.3.1 Some negative results

Here we show an example of a quite general elementary frame class on which all the modal memory logics we presented lack the Beth property.

Given a frame $\mathcal{F}$, the *connected components* of $\mathcal{F}$ are the maximal connected submodels of $\mathcal{F}$ (in the sense of graph theory, that is, taking the underlying undirected graph induced by $\mathcal{F}$). It is quite easy to see that the class of frames such that all the connected components are of size greater than 1 is first order definable with the formula $(\forall w)(\exists v)(w \neq v \wedge (wRv \vee vRw))$.

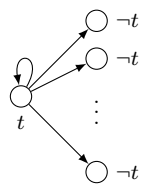
5.3.6. Theorem. *Let $\mathcal{C}$ be a class of frames such that all the connected components are of size greater than 1. Then $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ lacks the Beth property relative to $\mathcal{C}$.*

PROOF. Consider the following formulas:

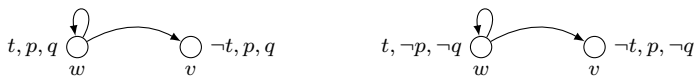
$$\begin{array}{ll} \neg ⓚ, & t \to \langle\langle r \rangle\rangle ⓚ, \\ t \to \langle\langle r \rangle\rangle \neg t, & t \to [\![r]\!] \neg t, \\ \neg t \to [\![r]\!] \bot, & p \to [\![r]\!] q, \\ \neg p \to [\![r]\!] \neg q. & \end{array}$$

Let Σ be the set defined by these formulas. Let's see first that in a model based on a frame in $\mathcal{C}$, Σ implicitly defines q in terms of p. Let us take any model $\mathcal{M} = \langle W, (R_r)_{r \in \mathrm{REL}}, V, S \rangle$ based on a frame in $\mathcal{C}$ such that $\mathcal{M} \models \Sigma$. Let w be any point in $\mathcal{M}$. First notice that by $\neg ⓚ$, $S = \emptyset$. If $w \in V(t)$, by $t \to \langle\langle r \rangle\rangle ⓚ$, wRw. Then, by $t \to [\![r]\!] \neg t$ and $t \to \langle\langle r \rangle\rangle \neg t$, w has at least one successor v such that $\neg t$ holds at v, and at every w-successor $\neg t$ holds. Furthermore, by $\neg t \to [\![r]\!] \bot$, every successor of w is a dead end.

The other case is that $w \notin V(t)$. By $\neg t \to [\![r]\!] \bot$, w has no successors. Since all the connected components in $\mathcal{M}$ has a size greater than 1, there must be a v such that vRw. By $\neg t \to [\![r]\!] \bot$ again, t must hold in v, and we are again in the previous case. Therefore, $\mathcal{M}$ looks like a collection of patterns with this shape:



Clearly, by $\neg p \to [\![r]\!] \neg q$ and $p \to [\![r]\!] q$, fixing p in the points where t holds, defines q for all the points in $\mathcal{M}$. That means that Σ implicitly defines q. Let's see now that there is no explicit definition of q. Let's take these two models



$\mathcal{M}_1$ $\mathcal{M}_2$

Clearly both models belong to $\mathcal{C}$ and globally satisfy Σ. Let's suppose that there is a formula δ such that q does not occur in δ and such that $\Sigma \models^{glo}_{\mathcal{C}} q \leftrightarrow \delta$. A trivial bisimulation argument shows that for any formula φ in which q does not occur, $\mathcal{M}_1, v \leftrightarrows \mathcal{M}_2, v$. Therefore δ cannot exist. □

5.3.7. COROLLARY. *All the memory logics we presented in Section 2.1.3 lack the Beth property with respect to $\mathcal{C}$.*

PROOF. This is quite easy to see. Since $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ is the weakest of the memory logics in terms of expressive power, one can translate Σ to every other memory fragment. Since all the memory logics presented in Section 2.1.3 have the generated submodel property, the bisimulation argument we used for $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ also works for all of them. □

5.4 Observations

We have not analyzed the interpolation property for hybrid memory logics. There is a general negative interpolation result in [tC05], that states that $\mathcal{HL}(@, \downarrow)$ is the least expressive logic with interpolation extending $\mathcal{HL}(@)$. Using what we have learnt from Chapter 3, we know that all the hybrid memory fragments we presented that extend $\mathcal{HL}(@)$ are strictly less expressive than $\mathcal{HL}(@, \downarrow)$. Therefore, we expect these hybrid memory fragments to be good candidates to fall into this negative result, and therefore lack interpolation over propositional symbols and nominals.

With respect to Beth, much work is left to be done. We would like to confirm (or disprove!) Conjecture 5.3.1. One way to try to prove this is to use the tableau system for $\mathcal{ML}^m(\langle\langle r \rangle\rangle)$ we are going to present in Chapter 7, and modify it to perform a model search for global satisfiability. We also want to explore more general classes of models for the remaining memory fragments, and determine the behavior of the Beth property there.

Chapter 6

Axiomatizations

Recuerda que las horas
se construyen
con los ladrillos
de tu alma de cristal.

"Vacío sideral", Luis Alberto Spinetta.

In this chapter we will present a number of axiomatizations and prove them (strongly) complete with respect to several memory logics. As we are going to see, nominals turn out to be a very useful tool to achieve completeness, so we will mostly focus in axiomatizations for hybrid memory logics.

Finding a sound and complete axiomatization for fragments where nominals are not present does not look easy to achieve, mainly because it seems hard to describe the interaction between the memory operators ⓡ and ⓚ and the modal operator $\langle r \rangle$. The difficulty of finding these axiomatizations lies on the fact that there is no easy way to name a point w without nominals, and therefore to describe the act of remembering w in terms of the other operators. This difficulty does not seem to be overcame with the additional expressive power that ⓔ or ⓕ provides.

On the other hand, nominals are a useful device we can use to accomplish this. The ability to use nominals to *name* points, and the use of techniques borrowed from classical hybrid modal logics allow us to obtain sound and strongly complete axiomatizations for several hybrid memory logic fragments. All the hybrid memory fragments we are going to analyze include the satisfaction operator @. The reason for this is due mainly for "hybrid reasons" rather than "memory reasons": the axiomatic system for $\mathcal{HL}$ (that is, the basic modal logic augmented with nominals *without* @) involves adding an infinite number of rules of proof (see [BT98] for details). On the other hand, including the @ operator produces a much nicer axiomatic system, that we take as a starting point to develop the axiomatizations we need.

In the last section of this chapter we are going to see that there are some cases where we can provide axiomatizations even in the absence of nominals. The trick to do this is to concentrate on fragments that have the tree model property, as shown in Theorem 4.1.3 and Corollary 4.1.6. This allows us to describe the effect of memorizing a point restricted just to a propositional level, since the expressive power of ⓡ in these cases is quite weak.

6.1 Completeness for $\mathcal{HL}^m(@, \langle r \rangle)$

This section is devoted to prove a completeness result for $\mathcal{HL}^m(@, \langle r \rangle)$. We are going to use the same techniques presented in [BdRV01] for the hybrid logic $\mathcal{HL}(@)$, and the ability to construct a canonical *named* model will allow us to generalize the result for pure extensions. We are going to give a precise definition of these concepts later.

Our axiomatization is shown in Figure 6.1. It is an extension of the axiomatization for $\mathcal{HL}(@)$ presented in [BdRV01].

Axioms:			
CT	All classical tautologies	*Intro*	$\vdash (i \wedge p) \to @_i p$
$K_@$	$\vdash @_i(p \to q) \to @_i p \to @_i q$	*Self-dual*$_@$	$\vdash @_i p \leftrightarrow \neg @_i \neg p$
$K_{[r]}$	$\vdash [r](p \to q) \to ([r]p \to [r]q)$	*Ref*	$\vdash @_i i$
Sym	$\vdash @_i j \leftrightarrow @_j i$	*Nom*	$\vdash (@_i j \wedge @_j p) \to @_i p$
Agree	$\vdash @_j @_i p \leftrightarrow @_i p$	*Back*	$\vdash \langle r \rangle @_i p \to @_i p$
Rem	$\vdash @_i(ⓡ\varphi \leftrightarrow \varphi[ⓚ/(ⓚ \vee i)])$		
Rules:			
MP	If $\vdash \varphi$ and $\vdash \varphi \to \psi$ then $\vdash \psi$	*Gen*$_{[r]}$	If $\vdash \varphi$ then $\vdash [r]\varphi$
Name	$\vdash j \to \varphi$ then $\vdash \varphi$ (j not in φ)	*Gen*$_@$	If $\vdash \varphi$ then $\vdash @_i \varphi$
Paste	If $\vdash (@_i \langle r \rangle j \wedge @_j \varphi) \to \psi$ then $\vdash (@_i \langle r \rangle \varphi) \to \psi$ ($j \neq i$ and j is not in φ or ψ)		
SortedSub$_1$	If $\vdash \varphi$ then $\vdash \varphi[p/\psi]$ for any $p \in$ PROP		
SortedSub$_2$	If $\vdash \varphi$ then $\vdash \varphi[i/j]$ for any $i, j \in$ NOM		

The expression $\varphi[a/b]$ is the result of uniformly replacing all occurrences of a in φ by b.

Figure 6.1: Axiomatization for $\mathcal{HL}^m(@, \langle r \rangle)$.

The axiom characterizing the behavior of the memory operator is *Rem*. All the others axioms and rules are from $\mathcal{HL}(@)$. Therefore, to show soundness of the axiomatization, we only have to look at this new axiom. Intuitively, the axiom says that, when standing in a point named by i, the act of remembering the current point is equivalent to increase the extension of ⓚ with i throughout the formula. Formally:

6.1.1. Lemma. *Let $\mathcal{M}$ be a model and $w \in \mathcal{M}$ such that $\mathcal{M}, w \models i$. Then, for all $v \in \mathcal{M}$, $\mathcal{M}[w], v \models \varphi$ iff $\mathcal{M}, v \models \varphi[ⓚ/(ⓚ \vee i)]$.*

Proof. By induction on φ. For the base case, if φ is a proposition symbol or a nominal, then since $\varphi = \varphi[ⓚ/(ⓚ \vee i)]$ we have $\mathcal{M}[w], v \models \varphi$ iff $\mathcal{M}, v \models \varphi$. For the ⓚ case we have to prove $\mathcal{M}[w], v \models ⓚ$ iff $\mathcal{M}, v \models ⓚ \vee i$.

$\Rightarrow$) Assume that $\mathcal{M}[w], v \models ⓚ$. If $v = w$, then $\mathcal{M}, v \models i$, and therefore $\mathcal{M}, v \models ⓚ \vee i$. If $v \neq w$, then $\mathcal{M}, v \models ⓚ$, and hence $\mathcal{M}, v \models ⓚ \vee i$.

$\Leftarrow$) Let's assume that $\mathcal{M}, v \models \textcircled{k} \vee i$. If $v = w$, then $\mathcal{M}[w], v \models \textcircled{k}$. On the other hand, if $v \neq w$, then we know that $\mathcal{M}[w], v \models \neg i$, and therefore $\mathcal{M}, v \models \textcircled{k}$. We conclude $\mathcal{M}[w], v \models \textcircled{k}$.

The conjunction, negation, diamond, @ and remember cases are straightforward, using the inductive hypothesis and the fact that the replacement operation $[\textcircled{k}/(\textcircled{k} \vee i)]$ distributes over $\wedge$, $\neg$, $\langle r \rangle$, @ and $\textcircled{r}$. □

6.1.2. Corollary. Rem *is sound over the class of all models.*

Proof. Take an arbitrary model $\mathcal{M}$ and let $w \in \mathcal{M}$ be such that $\mathcal{M}, w \models i$. By definition $\mathcal{M}, v \models @_i \textcircled{r} \varphi$ iff $\mathcal{M}[w], w \models \varphi$. Applying the previous lemma, this happens iff $\mathcal{M}, w \models \varphi[\textcircled{k}/(\textcircled{k} \vee i)]$ iff (by definition) $\mathcal{M}, v \models @_i \varphi[\textcircled{k}/(\textcircled{k} \vee i)]$. □

It is worth noting that having nominals in the language is a key feature to describe the $\textcircled{r}/\textcircled{k}$ interaction with modal operators, and the *Rem* axiom strongly uses this feature. The possibility to identify with a nominal the point in which a remember operation is taking place allows us to fully describe the behavior of this interaction.

We now turn to completeness. We will build a Henkin model using named maximal consistent sets (MCSs) for an arbitrary consistent set (see [BdRV01] for further details).

6.1.3. Definition. An MCS is *named* if and only if it contains a nominal. We call any nominal belonging to an MCS a *name* for that MCS. Also, if Γ is an MCS and i is a nominal, then we call $\{\varphi \mid @_i \varphi \in \Gamma\}$ *a named set yielded by* Γ. Furthermore, we say that a model is *named* if every point in the model is the denotation of some nominal (for all $w \in W$ there is some nominal i such that $V(i) = \{w\}$).

The idea behind the construction presented in [BdRV01] is that we can extract all the information we need to build a named canonical model from a single MCS. We start by noting that hidden inside any MCS there is a collection of named MCSs with a number of relevant properties:

6.1.4. Lemma. *Let Γ be an* MCS*. For every nominal i, let Δ_i be $\{\varphi \mid @_i \varphi \in \Gamma\}$. Then, (i) for every nominal i, Δ_i is an* MCS *that contains i; (ii) for all nominals i and j, if $i \in \Delta_j$, then $\Delta_i = \Delta_j$; (iii) for all nominals i and j, $@_i \varphi \in \Delta_j \leftrightarrow @_i \varphi \in \Gamma$; and (iv) if i is a name for Γ then $\Gamma = \Delta_i$.*

Proof. We only sketch the proof, the full details can be found in [BdRV01]. Claim (i) can be proved using *Ref* (to guarantee that $i \in \Delta_i$), *Gen*$_@$ and *Self-dual*$_@$ (to prove that Δ_i is an MCS). Claim (ii) is proved using *Sym* and *Nom*, Claim (iii) follows by *Agree*. And Claim (iv) is obtained by *Intro* and *Self-dual*$_@$. □

Given a consistent set of formulas Σ, we can always expand it to an MCS Σ^+ using the standard Lindenbaum's Lemma. The problem is that nothing guarantees that this MCS will be named. In addition, as we want to extract named MCSs from named sets yielded by Σ^+, we have to ensure that there are enough named MCSs to use as existential witnesses during the construction of the Henkin model. Here is where the *Name* and *Paste* rules are useful. Expanding the language with new nominals, the *Name* rule is going to solve our first problem, and the *Paste* rule solves the second. We call an MCS Γ *pasted* iff $@_i\langle r\rangle\varphi \in \Gamma$ implies that for some nominal j, $@_i\langle r\rangle j \wedge @_j\varphi \in \Gamma$. *Name* and *Paste* guarantee that any consistent set of formulas can be extended to a named and pasted MCS.

6.1.5. Lemma (Extended Lindenbaum Lemma). *Let $\mathcal{S} = \langle \text{PROP}, \text{NOM}, \text{REL}\rangle$ be a signature, let* NOM′ *be a countably infinite collection of nominals disjoint from* NOM*, and let $\mathcal{S}'$ be the signature obtained by extending $\mathcal{S}$ with* NOM′*. Then every $\mathcal{HL}^m(@, \langle r\rangle)$-consistent set of formulas in S can be extended to a named and pasted* MCS *in $\mathcal{S}'$.*

Proof. Full details can be found in [BdRV01]. The proof follows the standard Lindenbaum's construction with the following modifications. Take a consistent set of formulas Σ, and name it by adding a new nominal k (use *Name* to prove consistency). Using an enumeration of all the formulas, we expand Σ step-by-step with a formula that is consistent with the expanded set at each point. Because we want the final MCS to be pasted, at the $(m+1)$-th step, when we are considering Σ^m and the formula φ_{m+1}, if $\Sigma^m \cup \{\varphi_{m+1}\}$ is inconsistent, we set $\Sigma^{m+1} = \Sigma^m$. Else, if φ_{m+1} has the form $@_i\langle r\rangle\varphi$, we set $\Sigma^{m+1} = \Sigma^m \cup \{\varphi_{m+1}\} \cup \{@_i\langle r\rangle j \wedge @_j\langle r\rangle\varphi\}$, where j is new (relying on the *Paste* rule for consistency). If φ_{m+1} does not have the form $@_i\langle r\rangle\varphi$, we set $\Sigma^{m+1} = \Sigma^m \cup \{\varphi_{m+1}\}$ as usual. Finally, we take the infinite union of all the Σ^i. □

Now we can define the model we need, using the named sets yielded by a named and pasted MCS.

6.1.6. Definition. Let Γ be a named and pasted MCS. The *named model yielded by* Γ is $\mathcal{M}^\Gamma = (W^\Gamma, (R_r^\Gamma)_{r\in\text{REL}}, V^\Gamma, S^\Gamma)$. Here W^Γ is the set of all named sets yielded by Γ, $R_r^\Gamma(u, v)$ holds iff for all formulas φ, $\varphi \in v$ implies $\langle r\rangle\varphi \in u$, $V^\Gamma(a) = \{w \in W^\Gamma \mid a \in w\}$ for any atom a, and $S^\Gamma = \{w \mid ⓚ \in w\}$.

Note that $\mathcal{M}^\Gamma$ is well defined, since by items (i) and (ii) of Lemma 6.1.4, V^Γ assigns to every nominal a singleton subset of W^Γ. Using the fact that Γ is named and pasted, we can prove the following Existence Lemma:

6.1.7. Lemma (Existence Lemma). *Let Γ be a named and pasted* MCS*, and let $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ be the named model yielded by Γ. Suppose $u \in \mathcal{M}$ and $\langle r\rangle\varphi \in u$. Then there is a $v \in \mathcal{M}$ such that $R_r(u, v)$ and $\varphi \in v$*

Proof. The proof is similar as the one in [BdRV01]. As $u \in W$, for some nominal i we have that $u = \Delta_i$ (recall that $\Delta_i = \{\varphi \mid @_i\varphi \in \Gamma\}$). Hence as $\langle r \rangle \varphi \in u$, $@_i\langle r \rangle \varphi \in \Gamma$. But Γ is pasted so for some nominal j, $@_i\langle r \rangle j \wedge @_j\varphi \in \Gamma$, and so $\langle r \rangle j \in \Delta_i$ and $\varphi \in \Delta_j$. If now we show that $\Delta_i R \Delta_j$, then Δ_j would be a suitable choice for v. So suppose that $\psi \in \Delta_j$. This means that $@_j\psi \in \Gamma$. By @-agree (item (iii)) of Lemma 6.1.4 $@_j\psi \in \Delta_i$. But $\langle r \rangle j \in \Delta_i$. Hence, because $\langle r \rangle i \wedge @_i\phi \to \langle r \rangle \phi$ is provable with this axiomatization, $\langle r \rangle \psi \in \Delta_i$ as required. □

Now we are ready to prove the Truth Lemma that will lead us to the desired completeness result. Before that, to treat the ⓡ case properly, we have to redefine the complexity of the formulas, to be able to handle the substitutions made by the *Rem* axiom.

6.1.8. Definition. We define the complexity of a formula as $comp(\varphi) = 2(k + 1)(r + 1)(d + 1) + v$, where k, r and d are the number of occurrences of ⓚ, ⓡ and $\langle r \rangle$ respectively, and v is the number of occurrences of all the other possible operators.

Note that with this definition, $comp(ⓡ\varphi) > comp(\varphi[ⓚ/(ⓚ \vee i)])$.

6.1.9. Lemma (Truth Lemma). *Let $\mathcal{M} = \langle W, (R_r)_{r \in \mathrm{REL}}, V, S \rangle$ be the named model yielded by a named and pasted* MCS*, and let $u \in W$. Then, for all formulas φ, $\varphi \in u$ iff $\mathcal{M}, u \models \varphi$.*

Proof. By Induction on the complexity *comp* of φ. The atomic, boolean and modal cases are obvious (the Existence Lemma is used for the modal case, and the ⓚ case follows directly from the definition of S^Γ). We analyze the satisfaction operators. Suppose $\mathcal{M}, u \models @_i\psi$. This happens iff $\mathcal{M}, \Delta_i \models \psi$ (by items (i) and (ii) of Lemma 6.1.4, Δ_i is the only MCS containing i, and hence, by the atomic case of the present lemma, the only point in $\mathcal{M}$ where i is true) iff $\psi \in \Delta_i$ (by inductive hypothesis) iff $@_i\psi \in \Delta_i$ (using the fact that $i \in \Delta_i$ together with *Intro* for the left-to-right direction and *Intro* and *Self-dual*$_@$ for the right-to-left direction) iff $@_i\psi \in u$ (by *Agree*).

To finish the proof, let's analyze the case for ⓡ. Given $u \in \mathcal{M}$, we know that for some nominal i we have $u = \Delta_i$, so by definition, $\mathcal{M}, u \models i$ and $i \in u$. Suppose $\mathcal{M}, u \models ⓡ\psi$. This happens iff $\mathcal{M}, u \models @_i ⓡ\psi$ (because $\mathcal{M}, u \models i$) iff $\mathcal{M}, u \models @_i\psi[ⓚ/(ⓚ \vee i)]$ (by Corollary 6.1.2) iff $\mathcal{M}, u \models \psi[ⓚ/(ⓚ \vee i)]$ (again because $\mathcal{M}, u \models i$) iff $\psi[ⓚ/(ⓚ \vee i)] \in u$ (by inductive hypothesis) iff $@_i\psi[ⓚ/(ⓚ \vee i)] \in u$ (because $i \in u$, using *Intro* for the left-to-right direction, and *Self-dual*$_@$ and *Intro* for the right-to-left direction) iff $@_i ⓡ\psi \in u$ (by the *Rem* axiom) iff $ⓡ\psi \in u$ (because $i \in u$, applying again *Intro* and *Self-dual*$_@$). □

6.1.10. Theorem (Completeness for $\mathcal{HL}^m(@, \langle r \rangle)$). *Every* MCS *in the logic $\mathcal{HL}^m(@, \langle r \rangle)$ is satisfiable in a countable named model.*

Proof. Let Σ be a consistent set of formulas from $\mathcal{HL}^m(@, \langle r \rangle)$. We use the Extended Lindenbaum Lemma to expand it to a named and pasted set Σ^+ in an extended countable language. Let $\mathcal{M}$ be the named model yielded by Σ^+. By item (iv) of Lemma 6.1.4, because Σ^+ is named, Σ^+ is an element in the domain of $\mathcal{M}$. By the Truth Lemma, $\mathcal{M}, \Sigma^+ \models \Sigma$. The model is countable because each point is named by some nominal in the extended language, and there are only countably many of these. □

This establishes strong completeness as desired.

6.1.1 Pure extensions

As we anticipated in the introduction of this section, because our Henkin model is *named*, we can prove a more general result.

6.1.11. Definition. If a formula φ contains no propositional symbols (that is, its atoms are nominals or ⓚ), we say that φ is ⓚ*-pure*. Furthermore, if φ is a ⓚ-pure formula, we say that ψ is a ⓚ*-pure instance* of φ if ψ is obtained from φ by uniformly substituting nominals for nominals. A formula φ is *pure* if its atomic subformulas are only nominals.

The axiomatization we presented in Figure 6.1 for $\mathcal{HL}^m(@, \langle r \rangle)$ has the following property: for any set of pure formulas Π, if P is the logic obtained by adding the formulas in Π as axioms, then P is complete with respect to the class defined by Π.[1] This result can be extended to ⓚ-pure formulas for the case of $\mathcal{HL}^m_\emptyset(@, \langle r \rangle)$, the logic obtained over the class $\mathcal{C}_\emptyset$.

We first state a property that will help us achieve the completeness result for pure axioms.

6.1.12. Lemma. *Let $\mathcal{M} = \langle W, (R_r)_{r \in \mathrm{REL}}, V, S \rangle$ be a named model.*

1. *Let φ be a pure formula, and suppose that for all pure instances ψ of φ, $\mathcal{M} \models \psi$. Then for any V' and S', $\langle W, (R_r)_{r \in \mathrm{REL}}, V', S' \rangle \models \varphi$.*

2. *Let $S = \emptyset$ and φ be a ⓚ-pure formula. Suppose that for all ⓚ-pure instances ψ of φ, $\mathcal{M} \models \psi$. Then for any V', $\langle W, (R_r)_{r \in \mathrm{REL}}, V', S \rangle \models \varphi$.*

Proof. We only discuss item 2. Suppose that the hypothesis holds, but for some valuation V', $\langle W, (R_r)_{r \in \mathrm{REL}}, V', \emptyset \rangle \not\models \varphi$. We can take ρ, a ⓚ-pure instance of φ, such that ρ is obtained from φ replacing each nominal i by j, where $V'(i) = V(j)$. By an induction on the formula complexity, it is easy to see that $(W, (R_r)_{r \in \mathrm{REL}}, V, \emptyset) \not\models \rho$. This is a contradiction. □

[1] These general completeness results are standard in hybrid logics (see [BT98]).

With the help of Lemma 6.1.12, and since we showed that we can build named models from $\mathcal{HL}^m(@, \langle r \rangle)$-MCSs, a wide range of strong completeness results can be established (with the same proof as the one given in [BdRV01]).

6.1.13. THEOREM. *Let Π be a set of pure formulas and let $\mathcal{A}$ be the axiomatization obtained by adding formulas in Π as axioms to the axiomatization shown in Figure 6.1. Then, every $\mathcal{A}$-consistent set of formulas is satisfiable in a countable named model in the class defined by Π.*

PROOF. Given an $\mathcal{A}$-consistent set of formulas Ω, we can use the Extended Lindenbaum's Lemma to extend it to a named an pasted $\mathcal{A}$-MCS Ω^+. The named model $\mathcal{M}^\Omega$ that Ω^+ gives rise to will satisfy Ω at Ω^+. In addition, as every formula in Π belongs to every $\mathcal{A}$-MCS, we have that $\mathcal{M}^\Omega \models \Pi$. Therefore, by Lemma 6.1.12, $\mathcal{M}^\Omega$ is in the class of models defined by Π. □

To finish this section we will discuss an extension of the axiomatization presented above to characterize $\mathcal{HL}^m_\emptyset(@, \langle r \rangle)$.

6.1.14. THEOREM. *The system obtained by extending the axiomatization in Figure 6.1 with the axiom (Empty) $\vdash \neg ⓚ$ is sound and strongly complete for the logic $\mathcal{HL}^m_\emptyset(@, \langle r \rangle)$.*

PROOF. Soundness of *Empty* is obvious for the class of $\mathcal{HL}^m_\emptyset(@, \langle r \rangle)$-models. The completeness proof is as the one for $\mathcal{HL}^m(@, \langle r \rangle)$, but in addition, thanks to *Empty*, all maximal consistent sets Δ_i are such that $\neg ⓚ \in \Delta_i$. Therefore, the final model yielded by Γ, $\mathcal{M}^\Gamma = \langle W^\Gamma, (R^\Gamma_r)_{r \in \text{REL}}, V^\Gamma, S^\Gamma \rangle$, is such that $S^\Gamma = \emptyset$, and thus, it is a $\mathcal{HL}^m_\emptyset(@, \langle r \rangle)$-model. □

6.1.15. COROLLARY. *For the case of $\mathcal{HL}^m_\emptyset(@, \langle r \rangle)$, the result of adding Π, a set of pure formulas, can be extended to a set Π of ⓚ-pure formulas*

PROOF. Trivial, using Lemma 6.1.12, and the same proof as in Theorem 6.1.13. □

6.2 Dealing with other memory operators

We now turn to hybrid memory languages containing the ⓔ and ⓕ operators. We will first discuss completeness for $\mathcal{HL}^m(@, \langle r \rangle, ⓔ)$, then for $\mathcal{HL}^m(@, \langle r \rangle, ⓕ)$, and finally for the logic $\mathcal{HL}^m(@, \langle r \rangle, ⓔ, ⓕ)$.

6.2.1 Adding the ⓔ operator

We take as a starting point the axiomatization for $\mathcal{HL}^m(@, \langle r \rangle)$ presented in Figure 6.1. The first thing we should notice is that the *Rem* axiom is no longer sound. For example, take the valid formula $@_i ⓔ(ⓚ \vee i)$ and use *Rem* to conclude $@_i ⓡ ⓔ ⓚ$. This is clearly a contradiction, since after wiping out the memory, ⓚ cannot be true. Observe that the problem lays in the interaction between ⓡ and ⓔ. The replacement operation defined by *Rem* cannot be carried out throughout the whole formula: it should avoid replacements within the scope of an ⓔ. More formally, for each formula φ and nominal i we define the formula $\varphi_i^\star$ as follows:

$$
\begin{array}{rcl}
p_i^\star & = & p \quad p \in \text{PROP} \cup \text{NOM} \\
ⓚ_i^\star & = & ⓚ \vee i \\
(\neg\varphi)_i^\star & = & \neg\varphi_i^\star \\
(\varphi_1 \wedge \varphi_2)_i^\star & = & {\varphi_1}_i^\star \wedge {\varphi_2}_i^\star \\
(ⓡ\varphi)_i^\star & = & ⓡ\varphi_i^\star \\
(\langle r \rangle\varphi)_i^\star & = & \langle r \rangle\varphi_i^\star \\
(@_j\varphi)_i^\star & = & @_j\varphi_i^\star \\
(ⓔ\varphi)_i^\star & = & ⓔ\varphi
\end{array}
$$

Analogously to Lemma 6.1.1, we can use $(\cdot)^\star$ to characterize the behavior of the ⓡ operator and its interaction with the ⓔ operator.

6.2.1. Lemma. *Let $\mathcal{M}$ be a model and $w \in \mathcal{M}$ such that $\mathcal{M}, w \models i$. Then $\mathcal{M}, w \models ⓡ\varphi$ iff $\mathcal{M}, w \models \varphi_i^\star$.*

Axioms:
All the axioms from $\mathcal{HL}(@)$
Rem' $\vdash @_i(ⓡ\varphi \leftrightarrow \varphi_i^\star)$
Erase$_1$ $\vdash ⓔ\neg ⓚ$
Erase$_2$ $\vdash ⓔ s \leftrightarrow s \quad s \in \text{PROP} \cup \text{NOM}$
Erase$_3$ $\vdash ⓔ\neg p \leftrightarrow \neg ⓔ p$
Erase$_4$ $\vdash ⓔ(p \wedge q) \leftrightarrow (ⓔ p \wedge ⓔ q)$
Erase$_5$ $\vdash ⓔ\langle r \rangle p \leftrightarrow \langle r \rangle ⓔ p$
Erase$_6$ $\vdash ⓔ @_i p \leftrightarrow @_i ⓔ p$
Erase$_7$ $\vdash @_i(ⓔ ⓡ\varphi \leftrightarrow ⓔ\varphi_i^\star)$
Rules:
All the rules from $\mathcal{HL}(@)$

Figure 6.2: Axiomatization for $\mathcal{HL}^m(@, \langle r \rangle, ⓔ)$.

This result naturally suggests an axiom *Rem'* (shown in Figure 6.2) that replaces *Rem*. To characterize the ⓔ operator, we should first notice that it behaves globally and that it does not change the evaluation point. This implies

that there is no interaction between ⓔ and $\neg, \wedge, \langle r \rangle$ and @. To describe the interaction between ⓔ and ⓡ we can again make use of the operation $(\cdot)^\star$. The detailed axiomatization is in Figure 6.2.

Soundness of this axiomatization is straightforward. The completeness proof uses the same techniques introduced in Section 6.1. The proof of the Truth Lemma is carried out by induction in the complexity of the formula, and the new axioms handle the case for ⓔ by appropriately reducing the complexity in order to use the inductive hypothesis, as it is done in Lemma 6.1.9.

So now we can give the strong completeness result for $\mathcal{HL}^m(@, \langle r \rangle, ⓔ))$. The proof of this theorem follows exactly the same technique used in Theorem 6.1.10.

6.2.2. Theorem (Completeness for $\mathcal{HL}^m(@, \langle r \rangle, ⓔ))$. *Every maximal consistent set in the logic $\mathcal{HL}^m(@, \langle r \rangle, ⓔ)$ is satisfiable in a countable named model.*

Since it is clear that Lemma 6.1.12 still holds in $\mathcal{HL}^m(@, \langle r \rangle, ⓔ)$, and the canonical model we built is named, it is easy to see that one can also establish a stronger completeness result in terms of pure formulas for $\mathcal{HL}^m(@, \langle r \rangle, ⓔ)$, in the same way as stated in Theorem 6.1.13 and Theorem 6.1.14.

6.2.3. Theorem (Completeness for $\mathcal{HL}^m(@, \langle r \rangle, ⓔ))$. *Let Π be a set of pure formulas and let $\mathcal{A}$ be the axiomatization obtained by adding formulas in Π as axioms to the axiomatization shown in Figure 6.2. Then, every $\mathcal{A}$-consistent set of formulas is satisfiable in a countable named model in the class defined by Π.*

6.2.2 Adding the ⓕ operator

Let's consider an axiomatization for $\mathcal{HL}^m(@, \langle r \rangle, ⓕ)$. The main complication, compared with the case we just discussed, is that the ⓕ operator has a *local* behavior, and clearly depends on the point of evaluation. Hence, describing its interaction with the ⓡ operator will be more involved. We will require two rewriting functions $(\cdot)^r$ and $(\cdot)^f$. Using these two functions, we can obtain a very simple axiomatization of $\mathcal{HL}^m(@, \langle r \rangle, ⓕ)$ (see Figure 6.3).

Axioms:
All the axioms from $\mathcal{HL}(@)$
Rem $\vdash @_i(ⓡ\varphi \leftrightarrow \varphi_i^r)$
Forg $\vdash @_i(ⓕ\varphi \leftrightarrow \varphi_i^f)$
Rules:
All the rules from $\mathcal{HL}(@)$

Figure 6.3: Axiomatization for $\mathcal{HL}^m(@, \langle r \rangle, ⓕ)$.

For each formula $\varphi \in \mathcal{HL}^m(@, \langle r \rangle, ⓕ)$ and nominal i, we define the formula φ_i^r as follows:

$$
\begin{array}{rcl}
p_i^r & = & p \quad p \in \textsc{prop} \cup \textsc{nom} \\
\textcircled{\mathtt{k}}_i^r & = & (\textcircled{\mathtt{k}} \vee i) \\
(\neg\varphi)_i^r & = & \neg\varphi_i^r \\
(\varphi_1 \wedge \varphi_2)_i^r & = & ({\varphi_1}_i^r \wedge {\varphi_2}_i^r) \\
(\textcircled{\mathtt{r}}\varphi)_i^r & = & \textcircled{\mathtt{r}}\varphi_i^r \\
(\langle r\rangle\varphi)_i^r & = & \langle r\rangle\varphi_i^r \\
(@_j\varphi)_i^r & = & @_j\varphi_i^r \\
(\textcircled{\mathtt{f}}\varphi)_i^r & = & \textcircled{\mathtt{f}}((i \to \varphi) \wedge (\neg i \to \varphi_i^r))
\end{array}
$$

6.2.4. Lemma. *For every pointed model $(\mathcal{M}, w)$ such that $\mathcal{M}, w \models i$, and for all $v \in \mathcal{M}$, $\mathcal{M}[+w], v \models \varphi$ iff $\mathcal{M}, v \models \varphi_i^r$.*

Proof. By induction on φ. For the base case, if φ is a proposition symbol or a nominal, say a, then since $a_i^r = a$ we have $\mathcal{M}[+w], v \models a$ iff $\mathcal{M}, v \models a$. For the $\textcircled{\mathtt{k}}$ case we have to prove $\mathcal{M}[+w], v \models \textcircled{\mathtt{k}}$ iff $\mathcal{M}, v \models \textcircled{\mathtt{k}} \vee i$.

$\Rightarrow$) Assume that $\mathcal{M}[+w], v \models \textcircled{\mathtt{k}}$. If $v = w$, then $\mathcal{M}, v \models i$, and therefore $\mathcal{M}, v \models \textcircled{\mathtt{k}} \vee i$. If $v \neq w$, then $\mathcal{M}, v \models \textcircled{\mathtt{k}}$, and hence $\mathcal{M}, v \models \textcircled{\mathtt{k}} \vee i$.

$\Leftarrow$) Let's assume that $\mathcal{M}, v \models \textcircled{\mathtt{k}} \vee i$. If $v = w$, then $\mathcal{M}[+w], v \models \textcircled{\mathtt{k}}$. On the other hand, if $v \neq w$, then we know that $\mathcal{M}[+w], v \models \neg i$, and therefore $\mathcal{M}, v \models \textcircled{\mathtt{k}}$. We conclude $\mathcal{M}[+w], v \models \textcircled{\mathtt{k}}$.

Let's analyze the $\varphi = \textcircled{\mathtt{f}}\psi$ case. Suppose that $v = w$, therefore $\mathcal{M}[+w], w \models \textcircled{\mathtt{f}}\psi$ iff $\mathcal{M}[+w, -w], w \models \psi$ iff $\mathcal{M}[-w], w \models \psi$ iff (because $\mathcal{M}[-w], w \models i$) $\mathcal{M}[-w], w \models (i \to \psi) \wedge (\neg i \to \psi_i^r)$ iff (by definition of $\textcircled{\mathtt{f}}$) $\mathcal{M}, w \models \textcircled{\mathtt{f}}((i \to \psi) \wedge (\neg i \to \psi_i^r))$. On the other hand, suppose $w \neq v$. Therefore, $\mathcal{M}[+w], v \models \textcircled{\mathtt{f}}\psi$ iff $\mathcal{M}[+w, -v], v \models \psi$ iff (because v and w are different points) $\mathcal{M}[-v, +w,], v \models \psi$ iff (by inductive hypothesis) $\mathcal{M}[-v], v \models \psi_i^r$ iff (because $\mathcal{M}[-v], v \models \neg i$) $\mathcal{M}[-v], v \models (i \to \psi) \wedge (\neg i \to \psi_i^r)$ iff (by definition of $\textcircled{\mathtt{f}}$) $\mathcal{M}, v \models \textcircled{\mathtt{f}}((i \to \psi) \wedge (\neg i \to \psi_i^r))$

The conjunction, negation, diamond, @ and remember cases are straightforward, using the inductive hypothesis and the fact that the translation from φ to φ_i^r distributes over $\wedge$, $\neg$, $\langle r\rangle$, @ and $\textcircled{\mathtt{r}}$. $\square$

6.2.5. Corollary. *Let $\mathcal{M}$ by a model, and $w \in \mathcal{M}$. Then $\mathcal{M}, w \models @_i(\textcircled{\mathtt{r}}\varphi \leftrightarrow \varphi_i^r)$.*

In the same way, we can define a formula φ_i^f to deal with the $\textcircled{\mathtt{f}}$ case:

$$
\begin{array}{rcl}
p_i^f & = & p \quad p \in \text{PROP} \cup \text{NOM} \\
\textcircled{k}_i^f & = & (\textcircled{k} \wedge \neg i) \\
(\neg\varphi)_i^f & = & \neg\varphi_i^f \\
(\varphi_1 \wedge \varphi_2)_i^f & = & ({\varphi_1}_i^f \wedge {\varphi_2}_i^f) \\
(\textcircled{f}\varphi)_i^f & = & \textcircled{f}\varphi_i^f \\
(\langle r\rangle\varphi)_i^f & = & \langle r\rangle\varphi_i^f \\
(@_j\varphi)_i^f & = & @_j\varphi_i^f \\
(\textcircled{r}\varphi)_i^f & = & \textcircled{r}((i \to \varphi) \wedge (\neg i \to \varphi_i^f))
\end{array}
$$

6.2.6. Lemma. *For every pointed model $(\mathcal{M}, w)$ such that $\mathcal{M}, w \models i$, and for all $v \in \mathcal{M}$, $\mathcal{M}[-w], v \models \varphi$ iff $\mathcal{M}, v \models \varphi_i^f$.*

Proof. By induction on φ. The only cases that are worth analyzing are $\textcircled{k}$ and $\textcircled{r}\psi$. The other cases are equivalent to the proof of Lemma 6.2.4. For the $\textcircled{k}$ case we have to prove $\mathcal{M}[-w], v \models \textcircled{k}$ iff $\mathcal{M}, v \models \textcircled{k} \wedge \neg i$.

$\Rightarrow$) Assume that $\mathcal{M}[-w], v \models \textcircled{k}$. If $v = w$, this is an absurd, so $v \neq w$. Therefore $\mathcal{M}, v \models \textcircled{k}$, and hence $\mathcal{M}, v \models \textcircled{k} \wedge \neg i$.

$\Leftarrow$) Let's assume that $\mathcal{M}, v \models \textcircled{k} \wedge \neg i$. If $v = w$, then $\mathcal{M}, v \models i$, so this is an absurd. Therefore $v \neq w$, and then we know that $\mathcal{M}[-w], v \models \textcircled{k}$, and therefore $\mathcal{M}[-w], v \models \textcircled{k} \wedge \neg i$.

Let's analyze the $\varphi = \textcircled{r}\psi$ case. Suppose that $v = w$, therefore $\mathcal{M}[-w], w \models \textcircled{r}\psi$ iff $\mathcal{M}[-w, +w], w \models \psi$ iff $\mathcal{M}[+w], w \models \psi$ iff (because $\mathcal{M}[+w], w \models i$) $\mathcal{M}[+w], w \models (i \to \psi) \wedge (\neg i \to \psi_i^f)$ iff (by definition of $\textcircled{r}$) $\mathcal{M}, w \models \textcircled{r}((i \to \psi) \wedge (\neg i \to \psi_i^f))$. On the other hand, suppose $w \neq v$. Therefore, $\mathcal{M}[-w], v \models \textcircled{r}\psi$ iff $\mathcal{M}[-w, +v], v \models \psi$ iff (because v and w are different points) $\mathcal{M}[+v, -w,], v \models \psi$ iff (by inductive hypothesis) $\mathcal{M}[+v], v \models \psi_i^f$ iff (because $\mathcal{M}[+v], v \models \neg i$) $\mathcal{M}[+v], v \models (i \to \psi) \wedge (\neg i \to \psi_i^f)$ iff (by definition of $\textcircled{r}$) $\mathcal{M}, v \models \textcircled{r}((i \to \psi) \wedge (\neg i \to \psi_i^f))$ $\square$

6.2.7. Corollary. *Let $\mathcal{M}$ by a model, and $w \in \mathcal{M}$. Then $\mathcal{M}, w \models @_i(\textcircled{f}\varphi \leftrightarrow \varphi_i^f)$.*

Soundness of *Rem* and *Forg* in the axiomatization of $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{f})$ in Figure 6.3 are a direct consequence of Corollaries 6.2.5 and 6.2.7.

To achieve completeness, we first have to give an adequate notion of complexity of formulas in such a way that the Truth Lemma for this logic can be shown. As in section 6.1, we look for a function *comp* : FORMS $\to \mathbb{N}$ such that $comp(\textcircled{r}\varphi) > comp(\varphi[\textcircled{k}/(\textcircled{k} \vee i)])$. But in this setting, to account for the new axioms of Figure 6.3, we have stronger restrictions: we need to find a function such that $comp(\textcircled{r}\varphi) > comp(\varphi_i^r)$ and $comp(\textcircled{f}\varphi) > comp(\varphi_i^f)$. The complexity given in Definition 6.1.8 is not suitable because the lengths of φ_i^r and φ_i^f are much larger than the length of φ. We next show some upper bounds for the lengths of φ_i^r and φ_i^f and then we define a suitable complexity function.

Observe that some right-hand formulas in the definition of φ_i^r and φ_i^f are abbreviations of formulas using $\wedge$ and $\neg$ as the only boolean connectives. Having this in mind, it can easily be shown the following equalities concerning $|\varphi|$, the length of a formula φ:

$$\begin{array}{rcl}
|(\textcircled{f}\varphi)_i^r| & = & 15 + |\varphi| + |\varphi_r^r| \\
|(\textcircled{r}\varphi)_i^f| & = & 15 + |\varphi| + |\varphi_i^f| \\
|(\varphi \wedge \psi)_i^*| & = & 3 + |\varphi_i^*| + |\psi_i^*| \qquad \text{for } * \in \{r, f\} \\
|(\dagger\varphi)_i^*| & = & 1 + |\varphi_i^*| \qquad \text{for } \dagger \in \{\neg, \langle r\rangle, @_j\} \text{ and } * \in \{r, f\} \\
|\textcircled{k}_i^r| & = & 8 \\
|\textcircled{k}_i^f| & = & 6
\end{array}$$

It can be shown by induction on φ that $\max\{|\varphi_i^r|, |\varphi_i^f|\} \leq (|\varphi| + 7)^2$. Let $n_r(\varphi)$ denote the nesting depth of $\textcircled{r}$ in the formula φ, i.e. the maximum number of occurrences of $\textcircled{r}$ along the paths of the syntactic tree of φ. In the same way, let $n_f(\varphi)$ denote the nesting depth of $\textcircled{f}$ in φ. Observe that $n_r(\varphi) = n_r(\varphi_i^r)$ and $n_f(\varphi) = n_f(\varphi_i^f)$.

Let $c(\varphi) : \textsc{forms} \to \mathbb{R}$ be defined as

$$c(\varphi) = 2^{3(n_r(\varphi)+n_f(\varphi))} \cdot \log|\varphi|.$$

The reader may verify that $c(\textcircled{r}\varphi) > c(\varphi_i^r)$ and $c(\textcircled{f}\varphi) > c(\varphi_i^f)$. Furthermore, for all the subformulas ψ of a formula φ, $c(\psi)$ is strictly increasing in $|\psi|$. Therefore, $comp : \textsc{forms} \to \mathbb{N}$ defined as

$$comp(\varphi) = 2^{c(\varphi)} = |\varphi|^{2^{3(n_r(\varphi)+n_f(\varphi))}}$$

is a suitable complexity function.

With the complexity function properly defined, strong completeness follows using the same techniques introduced in Section 6.1. As for $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{e})$, it is easy to see that the result holds for any pure axiomatic extension.

6.2.8. Theorem (Completeness for $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{f})$). *Let Π be a set of pure formulas and let $\mathcal{A}$ be the axiomatization obtained by adding formulas in Π as axioms to the axiomatization shown in Figure 6.3. Then, every $\mathcal{A}$-consistent set of formulas is satisfiable in a countable named model in the class defined by Π.*

6.2.3 The operators $\textcircled{e}$ and $\textcircled{f}$ together

Finally, putting together the ideas from the previous two axiomatizations, we obtain a sound and complete axiomatization for $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{e}, \textcircled{f})$. The first step is to extend the definition of φ_i^r and φ_i^f to handle the case of $\textcircled{e}$:

$$\begin{array}{rcl}(\textcircled{e}\varphi)_i^r & = & \textcircled{e}\varphi \\ (\textcircled{e}\varphi)_i^f & = & \textcircled{e}\varphi\end{array}$$

Note that Lemmas 6.2.4 and 6.2.6 still hold. Now we only need to add the axioms we used to characterize ⓔ with minor changes. Observe that the complexity function defined in Subsection 6.2.2 is appropriate for this case also. The final axiomatization is shown in Figure 6.4.

Axioms:
All the axioms from $\mathcal{HL}(@)$
Rem $\vdash @_i(\textcircled{r}\varphi \leftrightarrow \varphi_i^r)$
Forg $\vdash @_i(\textcircled{f}\varphi \leftrightarrow \varphi_i^f)$
$Erase_1$ $\vdash \textcircled{e}\neg\textcircled{k}$
$Erase_2$ $\vdash \textcircled{e}s \leftrightarrow s \quad s \in \text{PROP} \cup \text{NOM}$
$Erase_3$ $\vdash \textcircled{e}\neg p \leftrightarrow \neg\textcircled{e}p$
$Erase_4$ $\vdash \textcircled{e}(p \wedge q) \leftrightarrow (\textcircled{e}p \wedge \textcircled{e}q)$
$Erase_5$ $\vdash \textcircled{e}\langle r\rangle p \leftrightarrow \langle r\rangle\textcircled{e}p$
$Erase_6$ $\vdash \textcircled{e}@_i p \leftrightarrow @_i\textcircled{e}p$
$Erase_7'$ $\vdash @_i(\textcircled{e}\textcircled{r}\varphi \leftrightarrow \textcircled{e}\varphi_i^r)$
$Erase_8$ $\vdash \textcircled{e}\textcircled{f}\varphi \leftrightarrow \textcircled{e}\varphi$
Rules:
All the rules from $\mathcal{HL}(@)$

Figure 6.4: Axiomatization for $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{e}, \textcircled{f})$.

6.2.9. Theorem (Completeness for $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{e}, \textcircled{f})$). *Let Π be a set of pure formulas and let $\mathcal{A}$ be the axiomatization obtained by adding formulas in Π as axioms to the axiomatization shown in Figure 6.4. Then, every $\mathcal{A}$-consistent set of formulas is satisfiable in a countable named model in the class defined by Π.*

6.3 The case for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$

In the previous section we mentioned the importance of nominals to describe the interaction between memory and modal operators. In this section we will show that if we restrict ourselves to use $\langle\langle r\rangle\rangle$ instead of $\langle r\rangle$, it is possible to define a sound and complete axiomatization where nominals can be avoided. The key ingredient is that in this logic we can describe the interaction between ⓡ and ⓚ at the propositional level. This is not a coincidence. Because this logic has the tree model property (see Theorem 4.1.3), we can assume that we evaluate $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas on trees, and since there are no cycles, the remember operator has no real effect beyond the current point.

Given a formula φ, we define the formula $\varphi^\sharp$ as the result of replacing all the occurrences of ⓚ that are in φ at modal depth zero by $\top$. Formally:

$$\begin{array}{rcl} p^\sharp & = & p \quad p \in \textsc{prop} \\ \textcircled{k}^\sharp & = & \top \\ (\neg\varphi)^\sharp & = & \neg\varphi^\sharp \\ (\varphi_1 \wedge \varphi_2)^\sharp & = & \varphi_1^\sharp \wedge \varphi_2^\sharp \\ (\textcircled{r}\varphi)^\sharp & = & \textcircled{r}\varphi^\sharp \\ (\langle\langle r\rangle\rangle\varphi)^\sharp & = & \langle\langle r\rangle\rangle\varphi \end{array}$$

6.3.1. Lemma. *$\mathcal{M}, w \models \textcircled{r}\varphi$ iff $\mathcal{M}, w \models \varphi^\sharp$.*

Proof. We proceed by induction. The case for ⓚ, the propositional symbols and boolean connectives are straightforward. We analyze the other cases. For the case $\varphi = \textcircled{r}\psi$. $\mathcal{M}, w \models \textcircled{r}\textcircled{r}\psi$ iff $\mathcal{M}, w \models \textcircled{r}\psi$ iff (by inductive hypothesis) $\mathcal{M}, w \models \psi^\sharp$ iff $\mathcal{M}, w \models (\psi^\sharp)^\sharp$ iff (by inductive hypothesis) $\mathcal{M}, w \models \textcircled{r}(\psi^\sharp)$ iff $\mathcal{M}, w \models (\textcircled{r}\psi)^\sharp$. For the case $\varphi = \langle\langle r\rangle\rangle\psi$. $\mathcal{M}, w \models \textcircled{r}\langle\langle r\rangle\rangle\psi$ iff (by definition) $\mathcal{M}[w], w \models \langle\langle r\rangle\rangle\psi$ iff (by definition of $\langle\langle r\rangle\rangle$) there is a $v \in \mathcal{M}$, $R_r(w, v)$ such that $\mathcal{M}[w], v \models \psi$ iff (by definition of $\langle\langle r\rangle\rangle$) $\mathcal{M}, w \models \langle\langle r\rangle\rangle\psi$ iff (by definition of $\sharp$) $\mathcal{M}, w \models (\langle\langle r\rangle\rangle\psi)^\sharp$. □

The axiomatization for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ (see Figure 6.5) is an extension of the axiomatization for the basic modal logic [BdRV01] (using $[\![r]\!]$, the dual of $\langle\langle r\rangle\rangle$, instead of $[r]$), plus the axiom $Rem_{\langle\langle r\rangle\rangle} \vdash \textcircled{r}\varphi \leftrightarrow \varphi^\sharp$.

Axioms:	
CT	All classical tautologies
$K_{[\![r]\!]}$	$\vdash [\![r]\!](p \to q) \to ([\![r]\!]p \to [\![r]\!]q)$
$Rem_{\langle\langle r\rangle\rangle}$	$\vdash \textcircled{r}\varphi \leftrightarrow \varphi^\sharp$
Rules:	
MP	If $\vdash \varphi$ and $\vdash \varphi \to \psi$ then $\vdash \psi$
$Gen_{[\![r]\!]}$	If $\vdash \varphi$ then $\vdash [\![r]\!]\varphi$
Sub	If $\vdash \varphi$ then $\vdash \varphi[p/\psi]$ for any $p \in \textsc{prop}$

Figure 6.5: Axiomatization for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$.

Soundness of $Rem_{\langle\langle r\rangle\rangle}$ follows from Lemma 6.3.1. Soundness for the rest of the axioms and rules is straightforward. We will prove completeness with respect to the class of acyclic models, and therefore for the class of all models. We will use a step-by-step construction. I.e., instead of building the entire canonical model, we will carry out a stepwise selection from MCSs of the canonical model of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ as our basic building blocks.[2]

[2]Alternatively, one can take the standard canonical model and then unravel it to obtain a tree, and therefore acyclic, model.

We define $\mathcal{M}^c = \langle W^c, (R^c_r)_{r\in\text{REL}}, V^c, S^c\rangle$, the $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ canonical model, in the usual sense (see [BdRV01] for details). That is, W^c is the set of all maximal consistent sets of formulas of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, $R^c_r(\Gamma,\Delta)$ iff for all $\varphi \in \Delta$, $\langle r\rangle\varphi \in \Gamma$, $\Gamma \in V^c(p)$ iff $p \in \Gamma$ and $S^c = \{\Gamma \mid \textcircled{k} \in \Gamma\}$.

6.3.2. Definition. A network $\mathcal{N} = \langle N, (R_r)_{r\in\text{REL}}, v\rangle$ is a triple where N is a countable non-empty set of elements, each R_r is a binary relation on N, and v is a function that maps elements in N to maximal consistent sets.

We say that a network is *coherent* if (C1) $\bigcup_{r\in\text{REL}} R_r$ defines an acyclic graph and (C2) $R^c_r(v(s), v(t))$ for all $s, t \in N$ such that $R_r(s,t)$. A network is *saturated* if whenever $\langle\langle r\rangle\rangle\psi \in v(s)$ for some $s \in N$, then there is a $t \in N$ such that $R_r(s,t)$ and $\psi \in v(t)$.

We want networks to play the role of models, so we have to check that we have imposed the right conditions on a network to achieve this.

6.3.3. Definition. Let $\mathcal{N} = \langle N, (R_r)_{r\in\text{REL}}, v\rangle$ be a network. We define the induced valuation $V_{\mathcal{N}}(p) = \{s \in N \mid p \in v(s)\}$, the induced set of memorized points $S_{\mathcal{N}} = \{s \in N \mid \textcircled{k} \in v(s)\}$, and the induced model $\mathcal{M}_{\mathcal{N}} = \langle N, (R_r)_{r\in\text{REL}}, V_{\mathcal{N}}, S_{\mathcal{N}}\rangle$. $\mathcal{F}_{\mathcal{N}} = \langle N, (R_r)_{r\in\text{REL}}\rangle$ is called the *underlying frame* of $\mathcal{N}$.

We are now ready to prove a Truth Lemma.

6.3.4. Lemma (Truth Lemma). *Let $\mathcal{N} = \langle N, (R_r)_{r\in\text{REL}}, v\rangle$ be a coherent and saturated network. Then, for all φ and $s \in N$,*

$$\mathcal{M}_{\mathcal{N}}, s \models \varphi \quad\leftrightarrow\quad \varphi \in v(s).$$

Proof. Before we prove this lemma, let us observe the following property: let $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, S\rangle$ be an acyclic model, and let $w, v \in W$ be such that $R_r(w,v)$. Then for all formulas φ, $\mathcal{M}[w], v \models \varphi$ iff $\mathcal{M}, v \models \varphi$.

We now proceed by induction on φ. The propositional case, the $\textcircled{k}$ case and the boolean cases are straightforward, given the definition of $\mathcal{M}_{\mathcal{N}}$. Let's suppose that $\mathcal{M}_{\mathcal{N}}, s \models \textcircled{r}\psi$. This happens iff (by Lemma 6.3.1) $\mathcal{M}_{\mathcal{N}}, s \models \psi^\sharp$ iff (by inductive hypothesis) $\psi^\sharp \in v(s)$ iff (by $Rem_{\langle\langle r\rangle\rangle}$ axiom) $\textcircled{r}\psi \in v(s)$.

The $\langle\langle r\rangle\rangle$ case: for the left-to-right direction, if $\mathcal{M}_{\mathcal{N}}, s \models \langle\langle r\rangle\rangle\psi$, then there exists $t \in N$ such that $R_r(s,t)$ and $\mathcal{M}_{\mathcal{N}}[s], t \models \psi$. Therefore, $\mathcal{M}_{\mathcal{N}}, t \models \psi$. By inductive hypothesis, $\psi \in v(t)$. Because the network is coherent, and $R_r(s,t)$, then $R^c_r(v(s), v(t))$, and we conclude $\langle\langle r\rangle\rangle\psi \in v(s)$. For the other direction, let's suppose that $\langle\langle r\rangle\rangle\psi \in v(s)$. Because the network is saturated, there is a $t \in N$ such that $\psi \in v(t)$ and $R_r(s,t)$. By inductive hypothesis, $\mathcal{M}_{\mathcal{N}}, t \models \psi$, so $\mathcal{M}_{\mathcal{N}}[s], t \models \psi$, and therefore by definition, $\mathcal{M}_{\mathcal{N}}, s \models \langle\langle r\rangle\rangle\psi$. $\square$

Summing up, we have reduced the problem of finding a model for an MCS Δ to a search for a coherent and saturated network for Δ. The idea here is to start with a coherent network and, one step at a time, remove the defects that are preventing the network from being saturated.

6.3.5. Definition. Let $\mathcal{N}$ be a network. We say that $\mathcal{N}$ has a saturation defect if there is a node $s \in N$ and a formula $\langle\langle r \rangle\rangle \psi \in v(s)$ such that there is not a $t \in N$, $R(s,t)$ and $\psi \in v(t)$.

Because a coherent network may have saturation defects, we have to say more about what is the meaning of repairing a defect. We are going to *extend* a network with a saturation defect with another where the defect is corrected.

6.3.6. Definition. Let $\mathcal{N}_0 = \langle N_0, R_0, v_0 \rangle$ and $\mathcal{N}_1 = \langle N_1, R_1, v_1 \rangle$ be two networks. We say that $\mathcal{N}_1$ extends $\mathcal{N}_0$ if $\mathcal{F}_{\mathcal{N}_0}$ is a subframe of $\mathcal{F}_{\mathcal{N}_1}$ and v_0 agrees with v_1 on N_0.

The following lemma states that a saturation defect of a finite coherent network can always be repaired.

6.3.7. Lemma (Repair Lemma). *Let $\mathcal{N}$ be a finite and coherent network with a saturation defect. Then there is a network $\mathcal{N}'$ extending $\mathcal{N}$ without that defect.*

Proof. Because $\mathcal{N}$ has a a saturation defect, there is a node $s \in N$ and a formula $\langle\langle r \rangle\rangle \psi \in v(s)$ such that there is not a $t \in N$, $R_r(s,t)$ and $\psi \in v(t)$. We define $\mathcal{N}'$ as

$$\begin{array}{rcl} N' & = & N \cup \{s'\} \quad \text{with } s' \notin N \\ R'_r & = & R_r \cup \{(s, s')\} \\ v' & = & v \cup \{(s', \Delta)\} \end{array}$$

where Δ is an MCS containing ψ such that $R^c_r(v(s), \Delta)$. We are going to prove the existence of such Δ through an Existence Lemma similar to Lemma 6.1.7 right away, so let's assume that for the moment. Clearly, $\mathcal{N}'$ is a coherent network extending $\mathcal{N}$ and does not have the previous defect. □

The Existence Lemma referred in the above proof is very similar to the one used for the basic modal logic in [BdRV01].

6.3.8. Lemma (Existence Lemma). *For all* MCS Γ *such that* $\langle\langle r \rangle\rangle \varphi \in \Gamma$, *there exists an* MCS Δ *such that* $\Gamma R^c \Delta$.

PROOF. Suppose $\langle\langle r\rangle\rangle\varphi \in w$. We will construct a point v such that wR^cv and $\varphi \in v$. Let v^- be $\{\varphi\} \cup \{\psi \mid [\![r]\!]\psi \in w\}$. Then v^- is consistent. For suppose not. Then there are $\psi_1, \ldots, \psi_n$ such that $\vdash (\psi_1 \wedge \cdots \wedge \psi_n) \to \neg\varphi$, and from this it is easy to see (using $K_{[\![r]\!]}$ and $Gen_{[\![r]\!]}$) that $\vdash [\![r]\!](\psi_1 \wedge \cdots \wedge \psi_n) \to [\![r]\!]\neg\varphi$. We conclude $\vdash ([\![r]\!]\psi_1 \wedge \ldots [\![r]\!] \wedge \psi_n) \to [\![r]\!]\neg\varphi$. Now, $[\![r]\!]\psi_1 \wedge \ldots [\![r]\!] \wedge \psi_n \in w$, therefore it follows that $[\![r]\!]\neg\varphi \in w$. Using that $[\![r]\!]\varphi$ is a rewriting of $\neg\langle\langle r\rangle\rangle\neg\varphi$, $\neg\langle\langle r\rangle\rangle\varphi \in w$. This is an absurd, since w is an MCS containing $\langle\langle r\rangle\rangle\varphi$.

By an easy argument, it is clear that wR^cv iff for all formulas φ, $[\![r]\!]\varphi \in w$ implies $\varphi \in v$. So let v be any MCS extending v^-. By construction, $\varphi \in v$. Furthermore, for all formulas ψ, $[\![r]\!]\psi \in w$ implies $\psi \in v$, and therefore wR^cv. □

Now we can prove the desired strong completeness result. We start with a singleton network, and we extend it step by step to a larger network using the Repair Lemma. We obtain the saturated network we want by taking the union of our sequence of networks.

6.3.9. THEOREM. *The axiomatization is strongly complete with respect to the class of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ models.*

PROOF. Let $S = \{s_i \mid i \in \omega\}$. Enumerate the potential saturation defects (the set $S \times$ FORMS). Given a consistent set Σ, expand it to an MCS Σ^+. The initial network is $\mathcal{N}^0 = \langle\{s_0\}, \emptyset, (s_0, \Sigma^+)\rangle$, which is finite and coherent. Given a network $\mathcal{N}^i$, $i \geq 0$, where the minimal saturation defect is D, we define $\mathcal{N}^{i+1}$ as the extension of $\mathcal{N}^i$ (following the Repair Lemma) without that defect. If $\mathcal{N}^i$ has no saturation defects, then $\mathcal{N}^{i+1} = \mathcal{N}^i$. Let $\mathcal{N}^\omega = \langle N, (R_r)_{r\in\text{REL}}, v\rangle$ be:

$$N = \bigcup_{n\in\omega} N^n \qquad R_r = \bigcup_{n\in\omega} R_r^n \qquad v = \bigcup_{n\in\omega} v^n.$$

It is clear that $\mathcal{N}^\omega$ is saturated. For suppose not; let d be the minimal saturation defect (with respect to the enumeration) of $\mathcal{N}^\omega$, say $d = d_k$. By construction, there must be an approximation $\mathcal{N}^i$ of $\mathcal{N}^\omega$ of which d is also a defect. There only can be k defects that are less than d, so d will be repaired before the stage $k + i$ of the construction. This is a contradiction, so $\mathcal{N}^\omega$ is a coherent and saturated network, and therefore $\mathcal{M}_{\mathcal{N}^\omega}, s_0 \models \Sigma$. □

6.3.1 An extension to $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$

Here we will make use of the equivalence preserving translation that takes formulas from $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ (as we saw in Section 4.1.1) to provide an axiomatization that includes the ⓕ operator.

The axiomatization for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ is shown in Figure 6.6. The operation $(\cdot)^\sharp$ is the one defined at the beginning of Section 6.3, that replaces all the occurrences of ⓚ that are in the applied formula at modal depth zero by $\top$, and

Axioms:	
CT	All classical tautologies
$K_{[\![r]\!]}$	$\vdash [\![r]\!](p \to q) \to ([\![r]\!]p \to [\![r]\!]q)$
$Rem^{ⓕ}_{\langle\langle r\rangle\rangle}$	$\vdash ⓡ\varphi \leftrightarrow \mathsf{Tr}_r(\varphi)^\sharp$
$Forg^{ⓕ}_{\langle\langle r\rangle\rangle}$	$\vdash ⓕ\varphi \leftrightarrow \mathsf{Tr}_f(\varphi)$
Rules:	
MP	If $\vdash \varphi$ and $\vdash \varphi \to \psi$ then $\vdash \psi$
$Gen_{[\![r]\!]}$	If $\vdash \varphi$ then $\vdash [\![r]\!]\varphi$
Sub	If $\vdash \varphi$ then $\vdash \varphi[p/\psi]$ for any $p \in \textsc{prop}$

Figure 6.6: Axiomatization for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$.

Tr is the equivalence preserving translation defined in Theorem 4.1.5 that takes formulas from $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$.

Soundness of this axiomatization is quite easy to establish. It is only worth mentioning the cases for $Rem^{ⓕ}_{\langle\langle r\rangle\rangle}$ and $Forg^{ⓕ}_{\langle\langle r\rangle\rangle}$.

6.3.10. Lemma. *The axioms* $\mathrm{Rem}^{ⓕ}_{\langle\langle r\rangle\rangle}$ *and* $\mathrm{Forg}^{ⓕ}_{\langle\langle r\rangle\rangle}$ *are sound.*

Proof. We have shown in Theorem 4.1.5 that for all φ in $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$, $\models \varphi \leftrightarrow \mathsf{Tr}_r(\varphi)$. Therefore, if we instantiate φ in $ⓡ\psi$, and we apply the definition of Tr_r, we get $\models ⓡ\psi \leftrightarrow ⓡ\mathsf{Tr}_r(\psi)$. Observe that $ⓡ\mathsf{Tr}_r(\psi)$ is an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formula, and therefore it is valid to apply the operation $(\cdot)^\sharp$ (defined for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas). Applying Lemma 6.3.1, we get $\models ⓡ\psi \leftrightarrow \mathsf{Tr}_r(\psi)^\sharp$ as desired. Soundness of $Forg^{ⓕ}_{\langle\langle r\rangle\rangle}$ follows directly from the definition of Tr and the fact that the translation preserves equivalence. □

To prove completeness we can reproduce the argument for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, and use a step-by-step model construction technique for the class of acyclic models. We only prove the Truth Lemma, since the other steps are identical.

6.3.11. Lemma (Truth Lemma). *Let* $\mathcal{N} = \langle N, (R_r)_{r\in\textsc{rel}}, v\rangle$ *be a coherent and saturated network. Then, for all* φ *and* $s \in N$,

$$\mathcal{M}_\mathcal{N}, s \models \varphi \quad \leftrightarrow \quad \varphi \in v(s).$$

Proof. We now proceed by induction on φ. We only prove the cases of ⓡ and ⓕ, since the other cases are identical as in Lemma 6.3.4. First, observe that both the translation Tr and the operation $(\cdot)^\sharp$ decrease the complexity of the formula.

Let's suppose that $\mathcal{M}_\mathcal{N}, s \models ⓡ\psi$. This happens iff (by Lemma 6.3.1) $\mathcal{M}_\mathcal{N}, s \models \mathsf{Tr}_r(\psi)^\sharp$ iff (by inductive hypothesis) $\mathsf{Tr}_r(\psi)^\sharp \in v(s)$ iff (by $Rem^{ⓕ}_{\langle\langle r\rangle\rangle}$ axiom) $ⓡ\psi \in v(s)$. For the ⓕ case. $\mathcal{M}_\mathcal{N}, s \models ⓕ\psi$ iff (by Lemma 6.3.1) $\mathcal{M}_\mathcal{N}, s \models \mathsf{Tr}_f(\psi)$ iff (by inductive hypothesis) $\mathsf{Tr}_f(\psi) \in v(s)$ iff (by $Forg^{ⓕ}_{\langle\langle r\rangle\rangle}$ axiom) $ⓕ\psi \in v(s)$. □

We conclude the desired result:

6.3.12. THEOREM. *The axiomatization is strongly complete with respect to the class of* $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ *models.*

6.4 Some final remarks

In this chapter we showed how nominals can be an effective tool to achieve completeness: by allowing to describe the precise interaction between $\textcircled{r}$ and $\textcircled{k}$ we could give a completeness result for $\mathcal{HL}^m(@, \langle r\rangle)$. Small variations of this axiomatization leads us to completeness results for other languages, as we showed for $\mathcal{HL}^m_\emptyset(@, \langle r\rangle)$, $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{e})$, $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{f})$ and $\mathcal{HL}^m(@, \langle r\rangle, \textcircled{e}, \textcircled{f})$. We also showed that in the case of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ and $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ we made use of the tree model property to overcome the lack of nominals. This is clearly not the case when we lose this property, as it is the case for $\mathcal{ML}^m(\langle r\rangle)$, $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{e})$, $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$, etc. It is not a coincidence then that the modal memory logics we proved undecidable in Chapter 4 are exactly the ones that are causing us trouble to find appropriate axiomatizations here, since those logics do not have the tree model property.

Being explicit, the remaining question for this chapter is the following:

6.4.1. QUESTION. Can we find sound and complete axiomatizations for memory logics without the tree model property, and which do not have nominals?

Following the same philosophy we kept throughout this work, our intention was to give the basic techniques to characterize memory operators using nominals, and not to exhaustively list all possible languages. Observe that, for example, the logic $\mathcal{HL}^m(\langle\langle r\rangle\rangle)$ can be easily axiomatized by replacing the *Back* axiom presented in Figure 6.1 by $\vdash @_i\langle r\rangle @_j\varphi \to @_j\varphi[\textcircled{k}/(\textcircled{k} \vee i)]$ (and similarly with the *Paste* rule).

Chapter 7

Tableau systems

Sacudido del árbol cayó
un fruto dulce muy raro

"El árbol del gran bonete", Patricio Rey y sus Redonditos de Ricota.

In the last chapter we presented axiomatic systems for different memory logic fragments, and we concentrated mainly in hybrid memory logics, since we saw that nominals were a key factor to achieve completeness. In this chapter we are going to turn to tableau systems for memory logics. We are going to use a *labeled* tableau system to be able to "name" points in the model, and we will see that having this ability will make nominals less essential here. Labels act as a meta-logical device to keep track of points, and in that sense they behave as nominals "outside" the language. As we said in Section 2.2.5, tableau systems can be regarded as a *backward reasoning* system, in the sense that they begin with the desired result and works backward from there to create a proof. In this way, they are closer to semantics than axiomatic systems, and they allow the possibility to have a finer-grained control on the derivation of a proof.

Here we are going to present a sound and complete tableau system for the logic $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$. The tableau works for both the class of all models and $\mathcal{C}_\emptyset$, the class of models with an empty memory. Tableau systems are in general quite modular, in the sense that each operator has an associated rule (or set of rules). This is not an exception in our case, and this modularity will allow us to drop rules to obtain sound and complete tableau systems for sublanguages of $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$.

As we proved in Chapter 4, $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ is not decidable, and therefore the tableau system we present is non-terminating. We are also interested in investigating the decidable memory fragments we found, and in providing a terminating tableau for those cases. Therefore, we are also going to present a terminating, sound and complete tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. This result is strongly based on the tree model property this logic enjoys, as we saw in Section 4.1.

We are not going to explore tableau systems for hybrid memory logics. However, this is quite straightforward to achieve taking the labeled tableau system presented in [BB07] for $\mathcal{HL}(@)$ and extending it with the memory rules we present in this chapter.

7.1 A tableau system for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$

The tableau rules we present here are an extension of the tableau system for the basic modal logic we showed in Section 2.2.5. To be able to present this system there are some concepts that need to be appropriately extended. First, we need to extend the idea of negated normal form we talked about in Section 2.2.5 to include memory operators. Since all the memory operators we defined are self-dual, the equivalence $\neg \bigcirc \varphi \equiv \bigcirc \neg \varphi$, with $\bigcirc \in \{ⓡ, ⓔ, ⓕ\}$, can be used to bring a formula into this form. Therefore, throughout this chapter we assume that formulas are in NNF.

The kind of formulas that can occur in a tableau branch are also an extended version of the one for the basic modal logic.

7.1.1. Definition. [Prefixed, accessibility and equality formulas] Let $W = \{w_1, w_2, \dots\}$ be an infinite, enumerable set of labels. Then $\langle w, R, F\rangle^C{:}\varphi$ is a *prefixed formula*, where $\varphi \in \mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$, $C \in \{\mathcal{C}_a, \mathcal{C}_\emptyset\}$, $w \in W$ and R and F are finite subsets of W. wR_rw' is an *accessibility formula* for $r \in \text{REL}$, and $w, w' \in W$. $w \approx w'$ is an *equality formula* for $w, w' \in W$.

Intuitively, in the prefix $\langle w, R, F\rangle^C$, w is the label of the point where the formula holds, C is the class of models we are working with ($\mathcal{C}_a$ is the class of all models, and $\mathcal{C}_\emptyset$ is the class of models with an empty memory). The set R is where we keep track of the explicitly memorized points, stored by ⓡ in the current branch. In the same way, F is the set of points that ⓕ explicitly forgets. Observe that since every prefixed formula is derived in finitely many steps, R and F will always be finite sets. The intended interpretation of wR_rw' is the same we used for the basic modal logic: the point denoted by w' is accessible from the point denoted by w by the interpretation of relation symbol r. Finally, the intended interpretation of an equality formula $w \approx w'$ is that w and w' label the same point in a given branch. We will use the term *formula* to denote either a formula of $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$, a prefixed formula, an accessibility formula, or an equality formula.

We present the set of rules in three different figures. In Figure 7.1 we show the classical rules for the basic modal logic, that handle propositional and modal cases. Observe that these rules do not modify the sets R and F, since their corresponding operators do not interact with the memory.

In Figure 7.2 we define the set of rules that deals with the memory. Let's try to grasp an intuitive idea of how they work. The general invariant that these rules assume is that R and F are always disjoint sets, and soundness of some of the rules depends on this. Rule (ⓚ) specifies that if w is in the memory, then either it is one of the explicitly memorized points, or it is in the initial memory, in which case ⓚ holds even with no explicitly memorized points. Rule (¬ⓚ) is the dual rule, and defines the interaction between ¬ⓚ and the set of explicitly forgotten points.

$$(\wedge)\quad \frac{\langle w,R,F\rangle^C{:}\varphi\wedge\psi}{\begin{array}{c}\langle w,R,F\rangle^C{:}\varphi\\ \langle w,R,F\rangle^C{:}\psi\end{array}} \qquad (\vee)\quad \frac{\langle w,R,F\rangle^C{:}\varphi\vee\psi}{\langle w,R,F\rangle^C{:}\varphi \mid \langle w,R,F\rangle^C{:}\psi}$$

$$(\langle r\rangle)\quad \frac{\langle w,R,F\rangle^C{:}\langle r\rangle\varphi}{\begin{array}{c}wR_rw'\\ \langle w',R,F\rangle^C{:}\varphi\end{array}}\ \dagger \qquad ([r])\quad \frac{\begin{array}{c}\langle w,R,F\rangle^C{:}[r]\varphi\\ wR_rw'\end{array}}{\langle w',R,F\rangle^C{:}\varphi}$$

Key:

† w' is fresh.

Figure 7.1: Propositional and modal rules

This rule uses the fact that R and F are disjoint, since if $\neg$ⓚ holds in w and w is not one of the explicitly forgotten points, then it is neither in the initial memory, nor in R. Rules for ⓡ and ⓕ simply store the current point in the appropriate set ((ⓡ) stores w in R, and (ⓕ) in F). These rules also make sure that R and F are always disjoint sets, removing w from the corresponding set. Rule (ⓔ) wipes out the explicitly memorized and forgotten points and evaluates the satisfiability of the formula in a model with no initial memory. Finally, the rule (repl) handles the equivalence relation between labels, and allow replacements among labels in the same equivalence class. Observe that the presence of the ⓔ modality may force the calculus to switch from the evaluation over C_a to that over $C_\emptyset$.

$$(ⓚ)\quad \frac{\langle w,\{v_1,\dots v_k\},F\rangle^C{:}ⓚ}{w\approx v_1 \mid \cdots \mid w\approx v_k \mid \langle w,\emptyset,\emptyset\rangle^C{:}ⓚ} \qquad (\neg ⓚ)\quad \frac{\langle w,R,\{v_1,\dots v_k\}\rangle^C{:}\neg ⓚ}{w\approx v_1 \mid \cdots \mid w\approx v_k \mid \langle w,\emptyset,\emptyset\rangle^C{:}\neg ⓚ}$$

$$(ⓕ)\quad \frac{\langle w,R,F\rangle^C{:}ⓕ\varphi}{\langle w,R-\{w\},F\cup\{w\}\rangle^C{:}\varphi} \qquad (ⓡ)\quad \frac{\langle w,R,F\rangle^C{:}ⓡ\varphi}{\langle w,R\cup\{w\},F-\{w\}\rangle^C{:}\varphi}$$

$$(ⓔ)\quad \frac{\langle w,R,F\rangle^C{:}ⓔ\varphi}{\langle w,\emptyset,\emptyset\rangle^{C_\emptyset}{:}\varphi} \qquad (\text{repl})\quad \frac{\begin{array}{c}\langle w,R,F\rangle^C{:}\varphi\\ w\approx^* w'\end{array}}{\langle w',R[w\mapsto w'],F[w\mapsto w']\rangle^C{:}\varphi}\ \ddagger$$

Key:

‡ $a \approx^* b$ iff (a,b) occurs in the reflexive, symmetric and transitive closure of the relation $\{(w,w') \mid w \approx w'$appears in the current branch$\}$. $A[w\mapsto w'] = A$ if $w\notin A$, and $(A-\{w\})\cup\{w'\}$ otherwise.

Figure 7.2: Memory rules

In Figure 7.3 we define the set of clash rules. Rule $(\bot_p)$ is the standard

clash rule for propositional symbols. Observe that the application of this rule only needs p and $\neg p$ to occur in the same point, regardless of the memory and the class of models. The rule $(\bot_{ⓚ})$ is the equivalent rule for ⓚ, but needs an empty memory, since otherwise the truth value of ⓚ could be modified by the current memory. Rules $(\bot_R)$ and $(\bot_F)$ relates ⓚ with the current memory: if w is explicitly memorized, then $\neg$ⓚ cannot hold. Equivalently, if w is explicitly forgotten, then ⓚ cannot hold. Note that $(\bot_R)$ also assumes that R and F are disjoint, since otherwise w could be memorized in R, but then forgotten by F. Finally, a branch can be immediately closed in the case ⓚ holds in an initially empty memory due to the rule $(\bot_\emptyset)$.

$$(\bot_p)\quad \frac{\langle w, R_1, F_1\rangle^{C_1}{:}p \qquad \langle w, R_2, F_2\rangle^{C_2}{:}\neg p}{\bot} \qquad\Bigg|\qquad (\bot_{ⓚ})\quad \frac{\langle w, \emptyset, \emptyset\rangle^{C}{:}ⓚ \qquad \langle w, \emptyset, \emptyset\rangle^{C}{:}\neg ⓚ}{\bot}$$

$$(\bot_R)\quad \frac{\langle w, \{w\}\cup R, F\rangle^{C}{:}\neg ⓚ}{\bot} \qquad\Bigg|\qquad (\bot_F)\quad \frac{\langle w, R, \{w\}\cup F\rangle^{C}{:}ⓚ}{\bot}$$

$$(\bot_\emptyset)\quad \frac{\langle w, \emptyset, \emptyset\rangle^{\mathcal{C}_\emptyset}{:}ⓚ}{\bot}$$

Figure 7.3: Clash rules

We are going to focus now on the completeness of the calculus. As usual, we will show that given an open and saturated branch Γ, we can define a model $\mathcal{M}^\Gamma$ that satisfies all the formulas that occur in the branch. To define the domain of $\mathcal{M}^\Gamma$ we first need the following definition.

7.1.2. Definition. [Eq_Γ] Let Γ be an open and saturated branch of a tableau for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$. Eq_Γ is the smallest equivalence relation extending $\{(w, w') \mid (w \approx w') \in \Gamma\}$.

7.1.3. Definition. [$\mathcal{M}^\Gamma$] Let Γ be an open and saturated branch of a tableau for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$. We define the induced model $\mathcal{M}^\Gamma = \langle W_\Gamma, (R_{r\Gamma})_{r\in\mathrm{REL}}, V_\Gamma, S_\Gamma\rangle$ as:

$$\begin{array}{lcl} W_\Gamma & = & \{w \mid w \text{ occurs in } \Gamma\}/\mathrm{Eq}_\Gamma \\ R_{r\Gamma} & = & \{([w],[w']) \mid wR_rw' \in \Gamma\} \\ V_\Gamma(p) & = & \{[w] \mid \langle w, R, F\rangle^C{:}p \in \Gamma, \text{ for any } R,\ F \text{ and } C\} \\ S_\Gamma & = & \{[w] \mid \langle w, \emptyset, \emptyset\rangle^{C_a}{:}ⓚ \in \Gamma\}, \end{array}$$

where $[w]$ is the equivalence class of w in Eq_Γ. Given a set of labels A, and a saturated and open branch Γ, we will denote $[A] = \{[v] \mid v \in A\}$. We call $\mathcal{M}^\Gamma_\emptyset = \langle W_\Gamma, R_\Gamma, V_\Gamma, \emptyset\rangle$, the version of $\mathcal{M}^\Gamma$ with an empty memory.

First we are going to show that if the tableau starts with R and F being disjoint, then this property holds for every prefixed formula at every branch.

7.1.4. Lemma. *Let Γ be an open and saturated branch of a tableau for the logic $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ that has as root a prefixed formula $\langle w, R, F\rangle^{\mathcal{C}}{:}\varphi$ such that $[R] \cap [F] = \emptyset$. Then, for every prefixed formula $\langle w', R', F'\rangle^{\mathcal{C}'}{:}\psi$ in Γ, $[R'] \cap [F'] = \emptyset$.*

Proof. We prove it by induction on the length of the derivation of Γ. The base case is obvious given the hypothesis. For the inductive step. The propositional and modal rules do not modify R and F, and the clash rules have no prefixed formulas as consequents. So let's analyze the memory rules. Both the prefixed consequents of the rules (ⓚ) and (¬ⓚ) and the consequent of the rule (ⓔ) set $R' = \emptyset$ and $F' = \emptyset$, so $R' \cap F' = \emptyset$ easily follows. The rules (ⓕ) and (ⓡ) delete w from one set and add w to the other set, so using the inductive hypothesis, $R' \cap F' = \emptyset$ is also easy to see. The only interesting case is the rule (repl). Let us assume, for the sake of contradiction, that $R[w \mapsto w'] \cap F[w \mapsto w'] \neq \emptyset$. Using the inductive hypothesis, we know that the only possibility is $R[w \mapsto w'] \cap F[w \mapsto w'] = \{w'\}$. If this rule was applied, we also know that $[w] = [w']$. This equivalence was introduced by the rules (ⓚ) or (¬ⓚ) (which are the only rules that introduce equivalences). Let's assume that $w \approx w'$ was previously introduced in the branch by (¬ⓚ). That means that $\langle w, R'', F''\rangle^{\mathcal{C}}{:}\neg$ⓚ $\in \Gamma$, for some R'' and F'', and $w' \in F''$. But then by (repl) and $\bot_F$, Γ cannot be open. The other case is that $w \approx w'$ was introduced by (ⓚ), and that means that $\langle w, R'', F''\rangle^{\mathcal{C}}{:}$ⓚ and $w' \in R''$. But then by (repl) and $\bot_R$, Γ cannot be open. Therefore, the only possibility for Γ to be an open branch is that $R[w \mapsto w'] \cap F[w \mapsto w'] = \emptyset$. □

Now we are ready to prove the desired lemma.

7.1.5. Lemma. *Let $\mathcal{M}^\Gamma = \langle W_\Gamma, (R_{r\Gamma})_{r\in\mathrm{REL}}, V_\Gamma, S_\Gamma\rangle$ be the induced model for Γ, where Γ is an open and saturated branch of a tableau for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ whose root $\langle w, R, F, :\rangle^C\varphi$ is such that $[R] \cap [F] = \emptyset$. Then:*

1. *$\langle w, R, F\rangle^{\mathcal{C}_a}{:}\varphi \in \Gamma$ implies $\mathcal{M}^\Gamma[+[R], -[F]], [w] \models \varphi$.*
2. *$\langle w, R, F\rangle^{\mathcal{C}_\emptyset}{:}\varphi \in \Gamma$ implies $\mathcal{M}^\Gamma_\emptyset[+[R], -[F]], [w] \models \varphi$.*

Proof. We proceed by induction on φ. We sometimes expand the sets R and F as $R = \{v_1, \ldots, v_k\}$ and $F = \{u_1, \ldots, u_j\}$.

Case $\varphi := p$. If $\langle w, R, F\rangle^{\mathcal{C}_a}{:}p \in \Gamma$, then $[w] \in V_\Gamma(p)$. Therefore, $\mathcal{M}^\Gamma[+[R], -[F]]$, $[w] \models p$. The case for $\mathcal{C}_\emptyset$ is analogous.

Case $\varphi := \neg p$. Suppose $\langle w, R, F\rangle^C{:}\neg p \in \Gamma$. If $\mathcal{M}^\Gamma[+[R], -[F]], [w] \models p$, it means that $[w] \in V_\Gamma(p)$ and hence, by definition, $\langle w, R', F'\rangle^{C'}{:}p \in \Gamma$ for some R', F' and C'. But in that case rule ($\bot_p$) applies and the branch would be closed. Again, the case for $\mathcal{C}_\emptyset$ is analogous.

Case $\varphi := ⓚ$**.** We consider the cases for $\mathcal{C}_a$ and $\mathcal{C}_\emptyset$:

1. If $\langle w, \{v_1, \ldots, v_k\}, \{u_1, \ldots, u_j\}\rangle^{\mathcal{C}_a}{:}ⓚ \in \Gamma$, then some consequent of the (ⓚ) rule must occur in Γ too. If $\langle w, \emptyset, \emptyset\rangle^{\mathcal{C}_a}{:}ⓚ \in \Gamma$, by definition, $[w] \in S_\Gamma$. If there is a $u_i \in \{u_1, \ldots, u_j\}$ such that $[w] = [u_i]$, then by (repl), $\langle u_i, \{v_1, \ldots, v_k\}[w \mapsto u_i], \{u_1, \ldots, u_j\}[w \mapsto u_i]\rangle^{\mathcal{C}_a}{:}ⓚ \in \Gamma$, and by $(\bot_F)$ the branch would be closed. Therefore, $[w] \notin \{[u_1], \ldots, [u_j]\}$ and hence, $\mathcal{M}[+[v_1], \ldots, +[v_k], -[u_1], \ldots, -[u_j]], [w] \models ⓚ$. The other case is that there is a v_i such that $[v_i] = [w]$. Therefore, to see $\mathcal{M}[+[v_1], \ldots, +[v_k], -[u_1], \ldots, -[u_j]], [w] \models ⓚ$, we only have to show that $[v_i] \notin \{[u_1], \ldots, [u_j]\}$, but this is guaranteed by Lemma 7.1.4.
2. If $\langle w, \{v_1, \ldots, v_k\}, \{u_1, \ldots, u_j\}\rangle^{\mathcal{C}_\emptyset}{:}ⓚ \in \Gamma$, then again some consequent of the (ⓚ) rule must occur in Γ too. The case $\langle w, \emptyset, \emptyset\rangle^{\mathcal{C}_\emptyset}{:}ⓚ \in \Gamma$ cannot occur, since by $(\bot_\emptyset)$ the branch would be closed. The other possibility is that there is a v_i such that $[v_i] = [w]$, and this case is analogous to case 1).

Case $\varphi := \neg ⓚ$**.** We consider again the cases for $\mathcal{C}_a$ and $\mathcal{C}_\emptyset$:

1. Assume $\langle w, \{v_1, \ldots, v_k\}, \{u_1, \ldots, u_j\}\rangle^{\mathcal{C}_a}{:}\neg ⓚ \in \Gamma$. Let's analyze the situation of $[w]$. If $[w] \in S_\Gamma$, using (repl) we can conclude that $\langle w, \emptyset, \emptyset\rangle^{\mathcal{C}_a}{:}ⓚ \in \Gamma$. Looking at the consequents of the $(\neg ⓚ)$ rule, $\langle w, \emptyset, \emptyset\rangle^{\mathcal{C}_a}{:}\neg ⓚ$ cannot be in the branch, since by $(\bot_{ⓚ})$ the branch would be closed. So the only possibility is that $w \approx u_i \in \Gamma$ for some $u_i \in \{u_1, \ldots, u_j\}$. That means $[w] = [u_i]$, and therefore $\mathcal{M}[+[v_1], \ldots, +[v_k], -[u_1], \ldots, -[u_j]], [w] \models \neg ⓚ$ as desired. Other case is that $[w] \in \{[v_1], \ldots, [v_k]\}$, but that means that there is a $v_i \in \{v_1, \ldots, v_k\}$ such that $[w] = [v_i]$. By (repl), $\langle v_i, \{v_1, \ldots, v_k\}[w \mapsto v_i], \{u_1, \ldots, u_j\}[w \mapsto v_i]\rangle^{\mathcal{C}_a}{:}\neg ⓚ \in \Gamma$ and by $(\bot_R)$ the branch would be closed. The last possibility is that $[w] \notin S_\Gamma$ and $[w] \notin \{[v_1], \ldots, [v_k]\}$. Therefore, $\mathcal{M}[+[v_1], \ldots, +[v_k], -[u_1], \ldots, -[u_j]]$, $[w] \models \neg ⓚ$.
2. If $\langle w, \{v_1, \ldots, v_k\}, \{u_1, \ldots, u_j\}\rangle^{\mathcal{C}_\emptyset}{:}\neg ⓚ \in \Gamma$, the proof is analogous to case 1).

Case $\varphi := ⓡ\psi$**.** Suppose $\langle w, R, F\rangle^{\mathcal{C}_a}{:}ⓡ\psi \in \Gamma$. By rule (ⓡ), we know $\langle w, R \cup \{w\}, F - \{w\}\rangle^{\mathcal{C}_a}{:}\psi \in \Gamma$. By inductive hypothesis, $\mathcal{M}^\Gamma[+[R \cup \{w\}], -[F - \{w\}]], [w] \models \psi$. Observe that $w \in (R \cup \{w\}) - (F - \{w\})$, and therefore $\mathcal{M}^\Gamma[+[R], -[F]], [w] \models ⓡ\psi$. The case for $\mathcal{C}_\emptyset$ is analogous.

Case $\varphi := ⓕ\psi$**.** Suppose $\langle w, R, F\rangle^{\mathcal{C}_a}{:}ⓕ\psi \in \Gamma$. By rule (ⓕ), we know $\langle w, R - \{w\}, F \cup \{w\}\rangle^{\mathcal{C}_a}{:}\psi \in \Gamma$. By inductive hypothesis, $\mathcal{M}^\Gamma[+[R - \{w\}], -[F \cup \{w\}]], [w] \models \psi$. Observe that $[w]$ is not in the current memory of $\mathcal{M}^\Gamma[+[R - \{w\}], -[F \cup \{w\}]]$, and therefore $\mathcal{M}^\Gamma[+[R], -[F]], [w] \models ⓕ\psi$. The case for $\mathcal{C}_\emptyset$ is analogous.

Case $\varphi := ⓔ\psi$**.** If $\langle w, R, F\rangle^{\mathcal{C}_a}{:}ⓔ\psi \in \Gamma$ then, by rule (ⓔ), $\langle w, \emptyset, \emptyset\rangle^{\mathcal{C}_\emptyset}{:}\psi \in \Gamma$ and, by inductive hypothesis, $\mathcal{M}^\Gamma_\emptyset, [w] \models \psi$. Therefore, it follows that $\mathcal{M}[+[R], -[F]], [w] \models ⓔ\psi$. The case for $\mathcal{C}_\emptyset$ is analogous.

The remaining boolean and modal cases are dealt with in the standard way. □

7.1.6. THEOREM. *The tableau calculus for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ is sound and complete for both the classes $\mathcal{C}_a$ and $\mathcal{C}_\emptyset$.*

More precisely, given $\varphi \in \mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$, φ is satisfiable iff any saturated tableau for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ with root $\langle w, \emptyset, \emptyset\rangle^{\mathcal{C}_a}{:}\varphi$ has an open branch. An equivalent result holds for the $\mathcal{C}_\emptyset$ class, starting with a tableau with root $\langle w, \emptyset, \emptyset\rangle^{\mathcal{C}_\emptyset}{:}\varphi$.

PROOF. Soundness is easy to see. Observe that the condition $[R] \cap [F] = \emptyset$ is fulfilled by the root of the tableau, and therefore using Lemma 7.1.4, by every prefixed formula in it. This is important to establish the soundness of ($\neg$ⓚ) and ($\bot_R$). Completeness is straightforward from Lemma 7.1.5: assume that a formula $\varphi \in \mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ is not satisfiable in the class C while there is a saturated tableau T with root $\langle w, \emptyset\rangle^C{:}\varphi$ and open branch Γ; $\mathcal{M}^\Gamma$ satisfies φ and is in the class C which contradicts the assumption. □

It is also straightforward to see that if we drop the (ⓔ) or (ⓕ) rules (or both) from the calculus, then we can prove soundness and completeness for the corresponding sublanguage of $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ with respect to both classes $\mathcal{C}_a$ and $\mathcal{C}_\emptyset$.

7.1.7. THEOREM. *The tableau calculus for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ without the (ⓔ) rule (the (ⓕ) rule) is sound and complete for $\mathcal{ML}^m(\langle r\rangle, ⓔ)$ ($\mathcal{ML}^m(\langle r\rangle, ⓕ)$) for both the classes $\mathcal{C}_a$ and $\mathcal{C}_\emptyset$. In the same way, the tableau calculus for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ without the (ⓔ) and (ⓕ) rules is sound and complete for $\mathcal{ML}^m(\langle r\rangle)$ for both the classes $\mathcal{C}_a$ and $\mathcal{C}_\emptyset$.*

7.2 A terminating tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$

In this section we are going to present a tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. As we saw in Section 4.1, $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ is decidable, and therefore we are interested in presenting a terminating tableau calculus. The calculus we present is again an extension of the tableau for the basic modal logic presented in Section 2.2.5. In this case, the notion of prefix is much simpler than the one we used for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$, since we do not have to keep track of the explicitly memorized and forgotten points. The tree model property is going to be a key ingredient, and we will only have to flag a point as memorized at a propositional level. Every time a modal transition is made, there is no need to keep that flag. Additionally, we do not need equality formulas anymore, since each label represents a point by itself, and no equivalence classes are needed.

7.2.1. Definition. [Prefixed and accessibility formulas] Let $W = \{w_1, w_2, \dots\}$ be an infinite, enumerable set of labels. Then $w_s{:}\varphi$ is a *prefixed formula*, where $\varphi \in \mathcal{ML}^m(\langle\langle r\rangle\rangle)$, s is either the flag R or the flag U, and $w \in W$. wR_rw' is an *accessibility formula* for $r \in \textsc{rel}$, and $w, w' \in W$.

Intuitively, w_R represents that the point w is explicitly memorized by ⓡ, and w_U means that the memorized state of w is unknown, in the sense that w could be or not in the initial memory of the model, but it is not explicitly memorized.

In Figure 7.4 we present the tableau rules for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. Rules $(\vee)$, $(\wedge)$, $(\bot_{ⓚ})$ and $(\bot_p)$ are just the same rules we used for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$, so we are not going to make further comments on them. Observe the interaction between the rules (ⓡ) and $(\bot_{ⓡ})$. The rule (ⓡ) flags the current point w as being memorized, and request that φ must be true there. Since w is currently memorized, $\neg$ⓚ cannot hold in w_R, and rule $(\bot_{ⓡ})$ watches for this. The fact that it is not necessary to carry a set of the memorized points, as we did in the previous section, is made explicit by the rules $(\langle\langle r\rangle\rangle)$ and $([\![r]\!])$. These rules takes a prefixed formula with label w, and independently of the flag w has, they flag w' with U. The intuitive meaning of this is that we are building a tree-like model, and therefore the effect of memorizing a point is overridden by the modal transitions.

$$(\wedge)\quad \frac{w_s{:}\varphi \wedge \psi}{\begin{array}{c} w_s{:}\varphi \\ w_s{:}\psi \end{array}} \qquad\qquad (\vee)\quad \frac{w_s{:}\varphi \vee \psi}{w_s{:}\varphi \mid w_s{:}\psi}$$

$$(\langle\langle r\rangle\rangle)\quad \frac{w_s{:}\langle\langle r\rangle\rangle\varphi}{\begin{array}{c} wR_rw' \\ w'_U{:}\varphi \end{array}}\ \dagger \qquad\qquad ([\![r]\!])\quad \frac{\begin{array}{c} w_s{:}[\![r]\!]\varphi \\ wR_rw' \end{array}}{w'_U{:}\varphi}$$

$$(ⓡ)\quad \frac{w_s{:}ⓡ\varphi}{w_R{:}\varphi} \qquad\qquad (\bot_{ⓚ})\quad \frac{\begin{array}{c} w_s{:}ⓚ \\ w_s{:}\neg ⓚ \end{array}}{\bot}$$

$$(\bot_{ⓡ})\quad \frac{w_R{:}\neg ⓚ}{\bot} \qquad\qquad (\bot_p)\quad \frac{\begin{array}{c} w_{s_1}{:}p \\ w_{s_2}{:}\neg p \end{array}}{\bot}$$

Key:

† w' is fresh.

Figure 7.4: Tableau rules for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$

In order to have a terminating tableau, we impose two general constraints on the construction of a tableau. Observe that these constraints are the same used by the tableau for the basic modal logic we showed in Section 2.2.5.

1. A formula is never added to a tableau branch where it already occurs.

2. The rule $(\langle\langle r\rangle\rangle)$ is never applied twice to a formula on a given branch.

We now define the notion of induced model of an open and saturated branch for this case.

7.2.2. Definition. $[\mathcal{M}^\Gamma]$ Let Γ be an open and saturated branch of a tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. Define the induced model $\mathcal{M}^\Gamma = \langle W_\Gamma, (R_{r\Gamma})_{r\in\text{REL}}, V_\Gamma, S_\Gamma\rangle$ as:

$$\begin{array}{lcl} W_\Gamma & = & \{w \mid w \text{ occurs in } \Gamma\} \\ R_{r\Gamma} & = & \{(w,w') \mid wR_rw' \in \Gamma\} \\ V_\Gamma(p) & = & \{w \mid w_s{:}p \in \Gamma, \text{ for } s \in \{U,R\}\} \\ S_\Gamma & = & \{w \mid w_U{:}\textcircled{k} \in \Gamma\}. \end{array}$$

The following lemma explicitly states that the induced model of an open and saturated branch is acyclic. Observe that together with the soundness and completeness proof we will show later, this can be regarded as an alternative for proving the tree model property we showed in Theorem 4.1.3.

7.2.3. Lemma. *Let $\mathcal{M}^\Gamma = \langle W_\Gamma, (R_{r\Gamma})_{r\in\text{REL}}, V_\Gamma, S_\Gamma\rangle$ be the induced model for Γ, where Γ is an open and saturated branch of a tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. Then $\mathcal{M}^\Gamma$ is acyclic.*

Proof. This property is quite easy to see. The only rule that adds pairs wR_rw' to the accessibility relationship is $(\langle\langle r\rangle\rangle)$, and each time it is applied, w' is requested to be new. In contrast with the induced model of $\mathcal{ML}^m(\langle r\rangle, \textcircled{e}, \textcircled{f})$, there is no equivalent class of points here, and each label is a point in $\mathcal{M}^\Gamma$ by itself. Therefore there cannot be a cycle in $\mathcal{M}^\Gamma$. □

Now we can prove the key lemma to show completeness.

7.2.4. Lemma. *Let $\mathcal{M}^\Gamma = \langle W_\Gamma, (R_{r\Gamma})_{r\in\text{REL}}, V_\Gamma, S_\Gamma\rangle$ be the induced model for Γ, where Γ is an open and saturated branch of a tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. Then:*

1. *$w_U{:}\varphi \in \Gamma$ implies $\mathcal{M}^\Gamma, w \models \varphi$.*

2. *$w_R{:}\varphi \in \Gamma$ implies $\mathcal{M}^\Gamma[w], w \models \varphi$.*

Proof. We proceed by induction on φ. The boolean cases $(\wedge)$ and $(\vee)$ are dealt with in the standard way.

Case $\varphi := p$**.** If $w_U{:}p \in \Gamma$, then $w \in V_\Gamma(p)$ and, therefore, $\mathcal{M}^\Gamma, w \models p$. The case for w_R is analogous using the fact that $\mathcal{M}, w \models p$ iff $\mathcal{M}[w], w \models p$.

Case $\varphi := \neg p$**.** Suppose $w_U{:}\neg p \in \Gamma$. If $\mathcal{M}^\Gamma, w \models p$, it means that $w \in V_\Gamma(p)$ and hence, by definition, $w_s{:}p \in \Gamma$ for $s \in \{R, U\}$. But in that case rule $(\bot_p)$ applies and the branch would be closed. Again, the case for w_R is analogous.

Case $\varphi := \textcircled{k}$**.** If $w_U{:}\textcircled{k} \in \Gamma$, then by definition $\mathcal{M}, w \models \textcircled{k}$. For the other case, it is always the case that $\mathcal{M}[w], w \models \textcircled{k}$.

Case $\varphi := \neg\textcircled{k}$**.** Suppose that $w_U{:}\neg\textcircled{k} \in \Gamma$, and assume for the sake of contradiction that $\mathcal{M}, w \models \textcircled{k}$. Then, by definition, $w_U{:}\textcircled{k} \in \Gamma$. By $(\bot_{\textcircled{k}})$, the branch would be closed. Therefore, $\mathcal{M}, w \models \neg\textcircled{k}$ as desired. For the other case, suppose that $w_R{:}\neg\textcircled{k} \in \Gamma$. By $(\bot_{\textcircled{r}})$, the branch would be closed, so this case cannot occur.

Case $\varphi := \textcircled{r}\psi$**.** Suppose $w_U{:}\textcircled{r}\psi \in \Gamma$. By rule $(\textcircled{r})$, we know $w_R{:}\psi \in \Gamma$. By inductive hypothesis, $\mathcal{M}^\Gamma[w], w \models \psi$. By definition, $\mathcal{M}^\Gamma, w \models \textcircled{r}\psi$. The case for w_R is identical.

Case $\varphi := \langle\!\langle r \rangle\!\rangle\psi$**.** Suppose $w_U{:}\langle r\rangle\psi \in \Gamma$. By rule $(\langle r\rangle)$, we know $w'_U{:}\psi \in \Gamma$ and wR_rw'. By inductive hypothesis, $\mathcal{M}^\Gamma, w' \models \psi$. Since $\mathcal{M}^\Gamma$ is acyclic, $\mathcal{M}, w \models \langle\!\langle r \rangle\!\rangle$. The case for w_R is identical, using again that $\mathcal{M}^\Gamma$ is acyclic.

Case $\varphi := [\![r]\!]\psi$**.** This case is analogous to $\varphi := \langle\!\langle r \rangle\!\rangle\psi$.

□

7.2.5. Theorem. *The tableau calculus for $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ is sound and complete for the class of all models.*

More precisely, given $\varphi \in \mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$, φ is satisfiable iff any saturated tableau for $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ with root $w_U{:}\varphi$ has an open branch.

Proof. Soundness is easy to see. Completeness is straightforward with the help of Lemma 7.2.4: assume that a formula $\varphi \in \mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ is not satisfiable in the class of all models, while there is a saturated tableau T with root $w_U{:}\varphi$ and open branch Γ; $\mathcal{M}^\Gamma$ satisfies φ which contradicts the assumption. □

7.2.1 Showing termination

Finally, we are going to show that every tableau derivation is finite. Since the tableau rules are all finitely branching, by König lemma [KÖ2], it suffices to see that the construction of every branch terminates, or in other words, that every saturated branch has finite length.

Given a prefixed formula $w_s{:}\varphi$, with s being the flag R or U, we call w_s the *prefix* of φ.

7.2.6. Definition. Given a tableau branch Γ and a prefix σ, the set of true formulas at σ on Γ, written $T_\Gamma(\sigma)$, is defined as $T_\Gamma(\sigma) = \{\varphi \mid \sigma{:}\varphi \in \Gamma\}$.

Notice that accessibility formulas are not included in $T_\Gamma(\sigma)$.

7.2.7. Lemma (Subformula Property). *Let T be a tableau with the prefixed formula $\sigma_0{:}\varphi_0$ as root. For any prefixed formula $\sigma{:}\varphi$ occurring on T, φ is a subformula of φ_0.*

Proof. This is easily seen by going through each of the tableau rules. □

7.2.8. Lemma. *Let Γ be a branch of a tableau, and let σ be any prefix occurring on Γ. Then the set $T_\Gamma(\sigma)$ is finite.*

Proof. Let $\sigma_0{:}\varphi_0$ denote the root formula of Γ. From Lemma 7.2.7, we know that $T_\Gamma(\sigma) \subseteq \{\varphi \mid \varphi \text{ is a subformula of } \varphi_0\}$, and hence $T_\Gamma(\sigma)$ is finite. □

7.2.9. Definition. Let T be a tableau. If a prefixed formula $\tau{:}\psi$ of T has been introduced by applying a rule to a premise $\sigma{:}\varphi$ of T then we say that $\tau{:}\psi$ is generated by $\sigma{:}\varphi$, and we write $\sigma : \varphi \prec \tau{:}\psi$.

7.2.10. Definition. Let Γ be a branch of a tableau. If a prefix τ has been introduced to the branch by applying a rule to a premise $\sigma{:}\varphi$ then we say that τ is generated by σ on Γ, and we write $\sigma \prec_\Gamma \tau$.

Now we define a measure for the complexity of a prefixed formula:

7.2.11. Definition. Let T be a tableau, $\sigma{:}\varphi$ be a prefixed formula occurring on T and let $|\varphi|$ denote the length of the φ. We define the complexity of $\sigma{:}\varphi$ just in terms of the length of φ, that is $m(\sigma{:}\varphi) = |\varphi|$.

Now we are going to show that the relationship $\prec$ defines a finitely branching tree, where each branch has finite length.

7.2.12. Lemma (Decreasing length). *Let T be a tableau. If $\sigma{:}\psi \prec \tau{:}\varphi$ then $m(\sigma{:}\psi) > m(\tau{:}\varphi)$.*

Proof. This is straightforward by going through each of the tableau rules. □

7.2.13. Lemma (Finite branching). *Let Γ be a branch of a tableau. For any $\sigma{:}\varphi \in \Gamma$ there is only a finite number of prefixed formulas $\tau{:}\psi \in \Gamma$ such that $\sigma{:}\varphi \prec \tau{:}\psi$.*

PROOF. For any given prefix σ the set $T_\Gamma(\sigma)$ is finite (Lemma 7.2.8). Additionally, for each formula φ in $T_\Gamma(\sigma)$, with the only exception of $(\langle\langle r\rangle\rangle)$, all the rules generates at most one new deterministic prefix when applied to $\sigma{:}\varphi$, since the first restriction imposed to the tableau construction requests that a formula is never added to a tableau branch where it already occurs. For the $(\langle\langle r\rangle\rangle)$ rule, the second restriction limits to one the number of applications to $\sigma{:}\varphi$, and therefore this rule generates exactly one new prefix for each formula. Thus, $\prec$ is finitely branching. □

7.2.14. THEOREM (TERMINATION). *Any tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ built using the rules presented in Figure 7.4 is finite.*

PROOF. Let $\sigma_0{:}\varphi_0$ be the root of a tableau T. Let's analyze the $\prec$ relationship of possible derivations of $\sigma_0{:}\varphi_0$. By Lemma 7.2.13, the derivation tree is finitely branching. By Lemma 7.2.12, each branch has finite length. By König Lemma, the derivation tree is finite. □

7.3 Tableau or axiomatic systems?

To close this chapter it is worth making some final remarks. We have seen that labeled tableau systems are a nice device for proof discovery. The ability to use labels overcame the difficulty we faced with axiomatic systems to describe the interaction between the memory operators ⓡ and ⓚ and the modal operator $\langle r\rangle$ without nominals.

Additionally, the tableau system we presented for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ could shed light on the task of discovering decidable (and interesting!) fragments of memory logics. The tableau rules we presented make explicit that a possible reason for undecidability is the unrestricted application of (repl). As it was shown in [AFGM09], limiting the number of applications of (repl) results in a terminating (although incomplete) tableau system. However, we could not find a reasonable fragment for which the restricted tableau is complete. One idea could be to restrict the class of models to the one where the number of cycles is bounded by a fixed k. Within this class, perhaps bounding the number of applications of (repl) to some number (that depends on k and the length of the input formula) would be enough to have a complete and terminating tableau for that class of models. The general question can be formulated as follows:

7.3.1. QUESTION. Could the tableau rules presented for $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$ be restricted in some way such that the resulting system is terminating and complete for some reasonable class of models?

Finally, the above comments could give the feeling that tableau systems are somehow "better" than axiomatic systems. As we said before, tableau systems

are closer to the semantics, and therefore sometimes the interaction between different operators is "hidden" in the derivation of a formula. On the other hand, axiomatic systems usually show these interactions explicitly, and some logic properties become clearer. For example, recall the axiomatization for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ we presented in Section 6.3. The axiom $Rem_{\langle\langle r\rangle\rangle}$ evidences that the effect of $\textcircled{\mathtt{r}}$ does not go beyond the propositional level. On the other hand, in the tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ we presented in Section 7.2, one may have to analyze the interaction of the rules $(\langle\langle r\rangle\rangle)$, $([\![r]\!])$ and $(\textcircled{\mathtt{r}})$ to notice that this property is present. Summing up, tableau and axiomatic systems are just two different ways (among others) to look at a characterization of a logic through a proof-theoretic device, and the advantages of each method depends on the context of use.

Chapter 8

Conclusions

–Vengo a venderle sus últimas palabras –dijo el hombre–. Son muy importantes porque a usted nunca le van a salir bien en el momento, y en cambio le conviene decirlas en el duro trance para configurar fácilmente un destino histórico retrospectivo.
–Traducí lo que dice- mandó el tiranuelo a su intérprete.
–Habla en argentino, Excelencia.
–¿En argentino? ¿Y por qué no entiendo nada?

"Cuento sin moraleja", Julio Cortázar.

We have taken a sort of "logic tour" in our quest for a better understanding of the memory logic family. We have visited different places, from which we have gained new perspectives to look at these modal fragments and learn how they behave in various situations. Like in any research work, we have left pending issues, and in many aspects we have only taken the first steps in our exploratory trip. We will continue working on this matter, to further expand our knowledge on memory logics.

This thesis can be seen as a contribution to the general enterprise of finding an appropriate logic for a specific task. The work we carried through can be regarded as a validation of the claim that modal logics make up a nice framework to combine different features, and that they offer a fine-grained set of tools that allow the logician to customize his own logical system. We have tried to show that there are many aspects that should be taken into consideration when designing a logic: expressive power and its complexity counterpart, the presence or absence of relevant meta-logical properties, and its characteristics from a proof-theoretical perspective.

We can also look at this work from the opposite perspective, and consider it as a way to measure the maturity of the modal logic tools that are currently available. Are really modal logics a plug-and-play system, in which a logician can choose between a menu of available features, understanding its pros and cons? Given a new modal logic family, as it was our case with memory logics, how developed are the tools to analyze it? How easy is this task? Are there standard procedures one can "run" on a logic to determine its properties? Of course, we do not have conclusive answers to these questions, but it is a good exercise to consider these concerns in the context of what we have done. Looking at the things we learned during this work, there are many useful "off-the-shelf" techniques to use, but the field of logic design is still young and there is a long

way to go. Much effort is needed to analyze particular cases, and the standard techniques only cover a part of the modal logic universe. Nevertheless, this is a very active and evolving area, and in the future one can expect that, step by step, more general questions are going to be answered.

8.1 A review of what we did

We started in Chapter 1 explaining why first order logic is not always *the* alternative to use, and that the existence of a big universe of logics is justified by different contexts of use, each of them with specific needs. For example, if we are in a situation where we need a feasible inference method, first order logic is probably not a good choice. Since expressive power and complexity are usually related, sometimes a less expressive logic is enough to model a particular problem, and we don't have to pay the extra cost in complexity. This is a possible real-case scenario for the applied computer science field, but there are other areas (like philosophy, for example) in which one can look for logics with certain theoretical properties, and where decidability is not of particular interest. The general aim of the first part Chapter 1 was to explain that our interpretation of "need" is quite general, and that we are interested both in practical and theoretical aspects of a logic.

In this context, we presented modal logics as a family of languages that have some characteristics in common: restricted quantification, characterizations of fragments of first and second order logic, and decidability in many of the cases. Modal logics can be regarded as a set of fine-grained tools for exploring the inner structure of classical systems, providing a wide option of modal operators to choose from. We presented some of them, in order to give an overview of the type of languages we were dealing with. We presented the basic modal logic, and some richer operators: the universal modality, since and until, and the ability to *name* points in a model through nominals, which led us to hybrid logics and binders. We then introduced memory logics, and the idea of adding an explicit memory to Kripke models. We gave further details about this family in Chapter 2, fixing a set as the storage structure. We showed some examples using ⓡ and ⓚ, the usual operators on sets that add elements and test membership. We also studied some variations within this schema: we proposed other operators, like ⓔ and ⓕ, we distinguished the class $\mathcal{C}_\emptyset$, a natural way to start evaluating a formula with an empty memory, and we looked at certain restrictions in the interplay between memory and modal operators, like the $\langle\langle r \rangle\rangle$ operator. In short, we showed that adding an explicit memory to Kripke models incorporates a new factor to consider, that can be used as a new ingredient to design a "made-to-fit" modal logic.

As we said before, this new family of logics can be seen as a way to model dynamic behavior through explicit memory operators that *change* the evaluating structure. But we can look at this approach from another perspective, that shows

that modal logic is an effective aid to think in new ways of *decomposing* operators which at first glance seem to be performing atomic actions. An example of this is the way hybrid logics helps to see the classical $\forall$ operator. One can reconsider the action that $\forall$ performs on a model as two separate things: the ability to *reach* all the elements in a model plus a way to *name* each of this elements. This can be modeled in hybrid logics with the universal modality A, plus the downarrow binder $\downarrow$, allowing the possibility to choose which specific aspects of $\forall$ we want in our custom logic. A similar thing occurs with memory logics, and we can break down the $\downarrow$ binder and the actions that it performs on the assignment function g from a similar perspective: g can be regarded as a sophisticated memory structure with unbounded size, direct access to all its memory cells, and where each stored element can be unequivocally retrieved. With this new insight, we can choose which type of storage we want to have, and therefore discover new ways of binding elements. Given that the addition of $\downarrow$ generally leads to undecidability, this perspective is particularly interesting.

In Chapters 3 and 4 we investigated the expressive power and the decidability of different combination of memory operators. To start the analysis we defined an appropriate notion of bisimulation, which allowed us to compare the expressivity among memory fragments. Bisimulations turned out to be a very useful tool, and the definition we proposed for each memory fragment ended up being a quite natural extension (once we understood them) of the bisimulation definition for the basic modal logic.

The results from Chapter 3 followed in general the intuitions: we have weakened the "storage structure" of $\downarrow$, and now having a set as memory, we cannot unequivocally retrieve elements anymore. Consequently, we proved that all the memory fragments we studied (up to $\mathcal{ML}^m(\langle r\rangle, ⓔ, ⓕ)$) are less expressive than $\mathcal{HL}(\downarrow)$. To be more precise, we could prove that $\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ) < \mathcal{HL}(\downarrow)$ and $\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ, ⓕ) \leq \mathcal{HL}(\downarrow)$, but we believe that $\mathcal{ML}^m_\emptyset(\langle r\rangle, ⓔ, ⓕ) \neq \mathcal{HL}(\downarrow)$ also holds, only that the models needed to show this are a little more involved. We also proved that the weakest memory fragment we defined, $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, is already strictly more expressive than the basic modal logic. We also analyzed other storage structures. It is quite obvious to see that using a list (and the usual operators to manipulate it) gives us the expressive power of $\downarrow$, so we wanted to try other structures. We analyzed then the case of using a stack. This result came as a surprise, since we expected to have a weaker logic than $\mathcal{HL}(\downarrow)$, but we proved that $\mathcal{ML}^{st}_\emptyset(\langle r\rangle) = \mathcal{HL}(\downarrow)$. The explanation can be found in the translation we used to encode $\mathcal{HL}(\downarrow)$ into $\mathcal{ML}^{st}_\emptyset(\langle r\rangle)$: we expected to "lose" elements when popping the top of the stack, but in fact the ability to "clone" the current stack gave us the capacity to keep a complete mapping of points. Summing up, the results we found in this section were quite promising, in the sense that adding a memory to Kripke models *does* make a difference in terms of expressivity. In this first exploratory trip, we found several fragments that allowed to gradually increase the expressive power, starting with the basic modal logic and ending up

with $\mathcal{HL}(\downarrow)$.

In terms of decidability, if we had to draw a general conclusion looking at the results shown in Chapter 4, we would say that although we have tamed the binding power of $\downarrow$ by changing the storage structure to a weaker one, expressivity is still very strong among memory logics. Most of the fragments turned out to be undecidable, and undecidability came in general with the ability to force cycles in a model. The examples of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ plus one single nominal and $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ are quite representative in that sense. The only decidable (in fact PSPACE-complete) fragments we found were $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ and $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \mathtt{f})$ (this last fragment turned out to be equivalent to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$). The key attribute we used to prove this was the tree model property of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, and this property was immediately lost when we forced the initial memory to be empty, or we added a nominal to the language. Furthermore, although $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ is strictly more expressive than the basic modal logic, the additional expressive power provided by $\mathtt{r}$ and $\mathtt{k}$ in this case was quite weak: the effect of $\mathtt{r}$ can be thought of as restricted just to the current evaluating point. This can be understood as a symptom that we have minimized the strength of $\mathtt{r}$ and $\mathtt{k}$, and that the contribution they make in terms of expressive power is not very relevant. In many ways, $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ is very much alike $\mathcal{ML}$, as we will discuss looking at the results found in Chapters 5, 6 and 7.

Chapter 4 is devoted to study Craig interpolation and Beth definability. We showed that interpolation fails for most of the memory fragments we defined. The technique we used to prove that a valid formula $\varphi \to \psi$ has an interpolant is quite standard. We reused the bisimilar models we had found in Chapter 3 and we adapted them to the common language of φ and ψ. The exception was $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, who has interpolation over propositional symbols and known. We followed here an adaptation of the technique presented in [tC05]. The main advantage of this technique is that it is purely model-theoretical, and does not need an axiomatic system (which we do not have, as we saw in Chapter 6). We could have used a tableau system instead, which we do have, but it was interesting to explore how $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ behaved with respect to some model theoretical concepts, such as ω-saturatedness and bisimulation products. We then analyzed Beth definability. Since interpolation and Beth are properties which are usually present (or absent) at the same time, we expected to have Beth for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. The problem we faced here was that the standard technique to prove Beth having interpolation fails, mainly because in the context of $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ the $[\![r]\!]$ operator memorizes the current point of evaluation before performing a modal transition. We could not find a proof for Beth for this fragment, but we stated a conjecture (that we believe is correct) from which the standard technique can be re-adapted. We also showed a quite general class of models where Beth definability fails for all the memory fragments we studied. There does not seem to be a standard way to disprove Beth definability, and definitively this aspect of memory logics deserves further analysis.

In Chapters 6 and 7 we analyzed memory logics from a proof theoretical perspective. We explored there axiomatizations and tableau characterizations for different memory fragment. These chapters show the main differences among both proof systems. While in axiomatic systems the interactions among operators is showed explicitly, it is not always easy to describe the behavior of an operator in terms of the other. This was case the case for $\mathcal{ML}^m(\langle r\rangle)$, where we could not find an axiomatic characterization. This problem was "cured" when we introduced nominals. The extra expressive power that nominals provide allowed to find sound and complete axiomatic system for all the hybrid memory fragments we defined. Even more, using standard hybrid techniques, we could prove automatic completeness for pure formulas (and ⓚ-pure formulas for the class $\mathcal{C}_\emptyset$). The tree model property was helpful again here, and we could also find a complete axiomatization for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, even without nominals. A complete axiomatic system for $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \text{ⓕ})$ was also easy to find, given the equivalence we found in Chapter 3 between $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \text{ⓕ})$ and $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$.

On the other hand, the prefixed labeled tableau systems we developed in Chapter 7 overcame the difficulty we faced with axiomatic systems to describe the interaction between the memory operators ⓡ and ⓚ and the modal operator $\langle r\rangle$ without nominals. As we already said in that chapter, this is mainly because labels act as "external" nominals in the tableau rules. We presented a unified tableau system for $\mathcal{ML}^m(\langle r\rangle, \text{ⓔ}, \text{ⓕ})$ that works for both the class of all models and $\mathcal{C}_\emptyset$. As it is usually the case with tableaus, there are specific rules for each logic operator, and therefore dropping the appropriate rules we could find sound and complete tableaus for sub-languages of $\mathcal{ML}^m(\langle r\rangle, \text{ⓔ}, \text{ⓕ})$. Given that all these logics are undecidable, these systems are of course non-terminating. We also presented a sound, complete and terminating tableau for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$, making strong use, again, of the tree model property. Both for $\mathcal{ML}^m(\langle r\rangle, \text{ⓔ}, \text{ⓕ})$ and $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ we extended the classic tableau system for the basic modal logic, but of course in the case of $\mathcal{ML}^m(\langle r\rangle, \text{ⓔ}, \text{ⓕ})$ we needed a more involved prefix, since there was more information we needed to carry throughout the derivation. Prefixed tableaus turned out to be a very flexible method to provide characterizations, and they adapted quite well to these logics.

8.2 How do we continue?

There are many pending issues to analyze, and as we said in the first part of this chapter, we are just at the beginning of our trip. We left some explicit open question during the development of this dissertation, many of them rather technical, and we are not going to repeat them here. We want instead to set up further lines of research from a more abstract perspective.

We said that memory logics are a particular type of dynamic logics, and that the idea of modifying a model when evaluating a formula is not new. Nevertheless,

memory logics does not bound the analysis to any fixed domain, and as we said before, most of the work that has been done in this direction implicitly adds some specific built-in behavior to the evolution of a model. Although we have connected memory logics with the basic modal logic and hybrid logics, we have not explored the connection with other dynamic logics. We want to examine this issue in order to "test" how general is our framework, and how flexible is to adapt to other type of needs where a specific domain is present. There are many types of dynamic logics to look for, but dynamic epistemic logics seem to be a good starting point. A possible connection may be in the line of the ideas we presented in Section 2.1.2, where we introduced an operator that every time we make a modal transition, all the points that satisfy some semantic condition are added to the memory. This could be a way to model epistemic states through a memory. There is also related work in [**?**]. Although different in spirit, the family of logics presented there study graph modifiers and some connections with logics of public announcements and hybrid logics.

There is also the decidability issue, which is always present from the computer scientist perspective. As we mentioned in the previous section, the restrictions we imposed to ⓡ and ⓚ to have decidability were perhaps overkilling, and other approaches can be examined. One line of research was suggested in Chapter 7, where we conjecture that restricting the class of models to one where models have a bounded (but fixed) number of cycles can led to a decidable fragment. In a more general way, analyzing different classes of models is always a good attempt, since it is possible to find very reasonable classes to work with, where decidability results can be established. Other line is to work with the storage structure. Although a set seems to be already a quite weak structure, we can think, for example, in restricting the set operators to allow to check whether a given point is in the memory, but to forbid knowing if a point is *not* memorized. This of course implies syntactic restrictions in the language. Finally, we can try to connect memory logics with the work with modal logics with concrete domains (like TPTL, see [AH89b], or description logics with concrete domains [Lut03]). The idea here is that instead of storing points of a model, one can store *values* that are associated to the points. This values can represent any concrete magnitude, like weight, temperature, time, etc. and the concrete part of the language provides operators to deal with this magnitudes and functions to map each point to its associated value. This idea can be regarded as another way of weakening the storage structure, given that values do not identify points, as many points can have the same associated value. In [AH89b] some decidability results have been established using the so-called *freeze* operator (that can be thought of as the version of $\downarrow$ that deals with values instead of points). Nevertheless, this result is achieved imposing strong restrictions on the Kripke model and the type of concrete domain to use. We can try to release some of those constraints by replacing the freeze operator with a memory that stores concrete magnitudes and appropriate operators to handle these values.

Bibliography

[ABM01] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: characterization, interpolation and complexity. *The Journal of Symbolic Logic*, 66(3):977–1010, 2001.

[ACD93] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[AFFM08] C. Areces, D. Figueira, S. Figueira, and S. Mera. Expressive power and decidability for memory logics. In *15th Workshop on Logic, Language, Information and Computation*, volume 5110 of *Lecture Notes in Computer Science*, pages 56–68, 2008.

[AFGM09] C. Areces, D. Figueira, D. Gorin, and S. Mera. Tableaux and model checking for memory logics. In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 5607 of *LNAI*, pages 47–61, Oslo, Norway, 2009. Springer Berling / Heidelberg. Proceedings of Tableaux09.

[AFH96] R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AFM09] C. Areces, S. Figueira, and S. Mera. Completeness results for memory logics. In Sergei Artemov and Anil Nerode, editors, *LFCS 09: Symposium on Logical Foundations of Computer Science*, volume 5407 of *LNCS*, pages 16–30, Deerfield Beach, FL, USA, 2009. Springer.

[AH89a] R. Alur and T. Henzinger. A really temporal logic. In *Journal of the ACM*, pages 164–169. IEEE Computer Society Press, 1989.

[AH89b] R. Alur and T.A. Henzinger. A really temporal logic. In *SFCS '89: Proceedings of the 30th Annual Symposium on Foundations of*

Computer Science, pages 164–169, Washington, DC, USA, 1989. IEEE Computer Society.

[All05] J.F. Allen. Towards a general theory of action and time. *The language of time: a reader*, page 251, 2005.

[AtC06] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*, pages 821–868. Elsevier, 2006.

[BB99] P. Battigalli and G. Bonanno. Recent results on belief, knowledge and the epistemic foundations of game theory. *Research in Economics*, 53(2):149–225, 1999.

[BB07] T. Bolander and P. Blackburn. Termination for hybrid tableaus. *Journal of Logic and Computation*, 17(3):517–554, 2007.

[BCPS07] F. Baader, D. Calvanese, and D.L. McGuinnessand D. Nardiand P.F. Patel-Schneider. *The description logic handbook*. Cambridge University Press New York, NY, USA, 2007.

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[Bet65] E.W. Beth. *The foundations of mathematics: a study in the philosophy of science*. North-Holland, 1965.

[BFB85] J. Barwise, S. Feferman, and J. Baldwin. *Model-theoretic logics*. Springer, 1985.

[Bla90] P. Blackburn. *Nominal tense logic and other sorted intensional frameworks*. PhD thesis, University of Edinburgh, Edinburgh, 1990.

[Bla00] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of IGPL*, 8(3):339–365, 2000.

[BS84] R. Bull and K. Segerberg. Basic modal logic. *1984*, pages 1–88, 1984.

[BS95] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995.

[BT98] P. Blackburn and M. Tzakova. Hybrid completeness. *Logic Journal of the IGPL*, 6(4):625–650, 1998.

[BWvB06] P. Blackburn, F. Wolter, and J. van Benthem, editors. *Handbook of Modal Logics*. Elsevier, 2006.

[Cre67] M.J. Cresswell. A henkin completeness theorem for t. *Notre Dame journal of formal logic*, 8:186–190, 1967.

[DGHP99] M. D'Agostino, D.M. Gabbay, R. Hähnle, and J. Posegga. *Handbook of tableau methods*. Kluwer Academic Publishers, 1999.

[Doe96] K. Doets. *Basic model theory*. University of Chicago Press, 1996.

[DR07] S. Demri and A. Rabinovich. The complexity of temporal logic with until and since over ordinals. In N. Dershowitz and A. Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2007.

[EFT84] H. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, 1984.

[Eme90] E.A. Emerson. Temporal and modal logic. *Handbook of theoretical computer science*, 1990.

[Esa74] L.L. Esakia. Topological Kripke models. In *Soviet Mathematics Doklady*, volume 15, pages 147–151, 1974.

[FHMV03] R. Fagin, J.Y. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. The MIT Press, 2003.

[Fin74] K. Fine. An incomplete logic containing S4. *Theoria*, 40(1):23–29, 1974.

[Fin75] K. Fine. Some connections between elementary and modal logic. In *Proceedings of the Third Scandinavian Logic Symposium*, pages 15–31, 1975.

[Fit72] M. Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, 13(2):237–247, 1972.

[Fit83] M. Fitting. *Proof methods for modal and intuitionistic logics*. D Reidel Pub Co, 1983.

[FL77] M.J. Fischer and R.E Ladner. Propositional modal logic of programs. In *Proceedings of the ninth annual ACM symposium on theory of computing*, pages 286–294. ACM New York, NY, USA, 1977.

[FL79] M.J. Fischer and R.E Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[Flo67] R. Floyd. Assigning meanings to programs. In *Proceedings of the American Mathematical Society Symposia on Applied Mathematics, Volume 19*, pages 19–31, 1967.

[Ger99] J. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, University of Amsterdam, 1999. ILLC Dissertation series DS-1999-01.

[GO05] V. Goranko and M. Otto. Model theory of modal logic. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of modal logic*, pages 249–329. Elsevier, 2005.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 163–173. ACM New York, NY, USA, 1980.

[GS91a] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[GS91b] J. Groenendijk and M. Stokhof. Two theories of dynamic semantics. In J. van Eijck, editor, *Logics in AI — European Workshop JELIA'90*, Lecture Notes in Artificial Intelligence, pages 55–64. Springer-Verlag, 1991.

[Hem96] E. Hemaspaandra. The price of universality. *Notre Dame Journal of Formal Logic*, 37(2):174–203, 1996.

[Hen90] T. Henzinger. Half-order modal logic: How to prove real-time properties. In *Proceedings of the Ninth Annual Symposium on Principles of Distributed Computing*, pages 281–296. ACM Press, 1990.

[Hin62] J. Hintikka. *Knowledge and belief*. Cornell Univ. Press Ithaca, New York, 1962.

[HLP90] E. Harel, O. Lichtenstein, and A. Pnueli. Explicit clock temporal logic. In *Proceedings of LICS'90*, pages 402–413, 1990.

[Hoa69] C. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–583, 1969.

[Hoo01] E. Hoogland. *Definability and interpolation: Model-theoretic investigations*. PhD thesis, Institute for Logic Language and Computation Universiteit van Amsterdam, 2001.

[HR06] I. Hodkinson and M. Reynolds. Temporal logic. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*, pages 655–720. Elsevier, 2006.

[JT51] B. Jónsson and A. Tarski. Boolean algebras with operators, part i. *American Journal of Mathematics*, 73:891–939, 1951.

[JT52] B. Jónsson and A. Tarski. Boolean algebras with operators, part ii. *American Journal of Mathematics*, 74:127–162, 1952.

[KÖ2] D. König. *Theorie der endlichen und unendlichen Graphen.* AMS Bookstore, 2002.

[Kam68] J.A.W. Kamp. *Tense logic and the theory of linear order.* PhD thesis, University of California, Los Angeles, 1968.

[Koy90] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[Kri63a] S.A. Kripke. Semantical Analysis of Modal Logic I. Normal Modal Propositional Calculi. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik*, 9, 1963.

[Kri63b] S.A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[Lad77] R.E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM journal on computing*, 6:467, 1977.

[Lew18] C.I. Lewis. *A Survey of Symbolic Logic.* University of California Press, 1918.

[LS77] E.J. Lemmon and D.S. Scott. *An Introduction to Modal Logic: The "Lemmon Notes".* Blackwell Publishing Ltd, 1977.

[Lut03] C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4.* King's College Publications, 2003.

[Mak66] D. Makinson. On some completeness theorems in modal logic. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik*, 12(113):379–384, 1966.

[Mar01] M. Marx. Tolerance logic. *Journal of Logic, Language and Information*, 10(3):353–374, 2001.

[McK45] J. C. C. McKinsey. On the syntactical construction of systems of modal logic. *Journal Symbolic Logic*, 10(3):83–94, 1945.

[MT44] J.C.C. McKinsey and A. Tarski. The algebra of topology. *Annals of mathematics*, 45(1):141–191, 1944.

[MT48] J. C. C. McKinsey and A. Tarski. Some theorems about the sentential calculi of Lewis and Heyting. *Journal Symbolic Logic*, 13(1):1–15, 1948.

[MV97] M. Marx and Y. Venema. *Multi-dimensional modal logic.* Kluwer Academic Pub, 1997.

[OW05] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *Proceedings of the 20th IEEE Symposium ofLogic in Computer Science (LICS 2005)*, pages 188–197, Chicago, IL, USA, June 2005. IEEE Comp. Soc. Press.

[Pla07] J. Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.

[Pnu77] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[Pra76] V.R. Pratt. Semantical consideration on Floyd-Hoare logic. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 109–121, 1976.

[Pra79] V.R. Pratt. Models of program logics. In *20th Annual Symposium on Foundations of Computer Science*, pages 115–122, 1979.

[Pri57] A.N. Prior. *Time and modality.* Oxford University Press, 1957.

[Pri67] A.N. Prior. *Past, present and future.* Oxford University Press, USA, 1967.

[Pri77] A. N. Prior. Papers on time and tense. *Oxford University Press*, 1977.

[Rey03] M. Reynolds. The complexity of the temporal logic with "until" over general linear time. *Journal of Computer and System Sciences*, 66(2):393–426, 2003.

[RV01] A. Robinson and A. Voronkov, editors. *Handbook of automated reasoning.* Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 2001.

[Sah73] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logics. In *Proceedings of the 3rd Scandinavian Logic Symposium*, volume 82, 1973.

[She83] V.B. Shehtman. Modal logics of domains on the real plane. *Studia Logica*, 42(1):63–80, 1983.

[Smu95] R.M. Smullyan. *First-order logic*. Dover Publications, 1995.

[Tar38] A. Tarski. Der aussagenkalkul und die topologie. *Fundamenta Mathematicae*, 31:103–134, 1938.

[tC05] B. ten Cate. *Model theory for extended modal languages*. PhD thesis, University of Amsterdam, 2005. ILLC Publications, Ph. D. Dissertation series, Amsterdam.

[Tho72] S.K. Thomason. Semantic analysis of tense logics. *Journal of Symbolic Logic*, pages 150–158, 1972.

[Tho74] S.K. Thomason. An incompleteness theorem in modal logic. *Theoria*, 40(1):30–34, 1974.

[Var97] M. Vardi. Why is modal logic so robustly decidable. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 31:149–184, 1997.

[vB84] J. van Benthem. Modal correspondence theory. *Handbook of Philosophical Logic*, 2:167–247, 1984.

[vB85] J. van Benthem. *Modal logic and classical logic*. Bibliopolis, 1985.

[vB91] J. van Benthem. *The logic of time*. Kluwer Academic Publishers, 1991.

[vB01] J. van Benthem. Logics for information update. In *TARK'01: Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 51–67. Morgan Kaufmann Publishers Inc., 2001.

[vB05] J. van Benthem. An essay on sabotage and obstruction. In *Mechanizing Mathematical Reasoning*, pages 268–276, 2005.

[vB06] J. van Benthem. One is a lonely number: on the logic of communication. In *Logic Colloquium*, volume 2, pages 96–129. Citeseer, 2006.

[vBL07] J. van Benthem and F. Liu. Dynamic logic of preference upgrade. *Journal of Applied Nonclassical Logics*, 17(2):157, 2007.

[vBvEK06] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.

[vDvdHK07] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Kluwer academic publishers, 2007.

[vEB97] P. van Emde Boas. The convenience of tilings. *Complexity, Logic and Recursion Theory*, 187:331–363, 1997.

[vW51] G.H. von Wright. An essay in modal logic. *North-Holland Publishing Company*, 1951.