Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía y Física

Relation-Changing Modal Logics

Thesis presented on December 16, 2013
to obtain

Doctorado en Ciencias de la Computación

Raul Alberto Fervari

Advisor

Dr. Carlos Areces	FaMAF, Universidad Nacional de Córdoba, Argentina

Jury

Dr. Javier Blanco	FaMAF, Universidad Nacional de Córdoba, Argentina
Dr. Hans van Ditmarsch	LORIA, CNRS - Université de Lorraine, France
Dr. Pedro Sánchez Terraf	FaMAF, Universidad Nacional de Córdoba, Argentina

Córdoba, Argentina, 2014

Contents

Acknowledgments

Almost four years ago I decided to start my PhD studies, and today I still think it was a good decision. I arrived at FaMAF in June, 2010 to meet Carlos and start working in Logic. The first ideas about Swap Logic came at the end of 2010, and later new ideas developed (which guided me to this thesis). Besides logic, I learned many other things from the trips I made and the people I met during my PhD studies. And precisely, this introductory text in my thesis is about people. It is about those who were with me from the beginning, and other that I have met thanks to the road I decided to take. I would like to thank all of them.

First, I would like to thank Carlos Areces for his supervision. Thanks for teaching me so many things. Thanks for all the time he dedicated to this thesis. Also, thanks for his advice, help and support (specially in moments of panic! :D) during these years. Thanks for introducing me to a lot of interesting people, and for sharing many things with me.

I am specially grateful to Guillaume Hoffmann for working with me since he arrived in Argentina. Thanks for his interest in the topics I was working on, and for collaborating in various results of this thesis. Thanks also, of course, for all the chats, drinks, the time we spent in Nancy, and many other things. I am also very grateful to Mauricio Martel for working with me on the undecidability results, for being available to travel to Córdoba as many times as needed, and for his enthusiasm.

I would like to give my thanks to Hans van Ditmarsch and François Schwarzentruber. Thanks for all the interesting ideas we discussed during my visit to LORIA, some of them included in the last part of this thesis. Thanks for all the hours of work, and for sharing new ideas to investigate in the future. A particular thank to Hans for accepting to be part of my jury.

Thanks to all the people in FaMAF that have heard me complaining about writing my thesis during several months, specially during lunch time (Pedro, Damián, Franco, Leti, Silvia, Miguel, and specially Chun), and to Eze, for the chats in the hall. Thanks to the Logics, Interaction and Intelligent Systems Group (LIIS), our group. Thanks to Seba ("the philosopher"), for some bibliography and for the Borges quote in the last chapter. Of course thanks to Javier and Pedro ST for accepting to be part of this thesis jury.

Thanks to all the people from LORIA who received me in Nancy, specially to the Argentinian guys (Juan and César). Thanks for making me feel comfortable there, for the meetings, the food, and for all the time in the music room (even on weekends!). Thanks to César for receiving me at his home in Nancy, and for finally visiting me in Córdoba. Thanks to the people I met at ESSLLI'12 (Facu, Nacho, Gustavo, Inari, etc.) for a great time in Opole.

Besides the people playing an important role in the academic part of my PhD studies, there are people from the other part, my friends since always. I want to thank all of them: friends from my town, from the university, and those I have met for no reason.

Finally, thanks to my family for all their support in every decision I take.

Raul Fervari
Córdoba, Argentina
April, 2014

Abstract

In this thesis we study dynamic modal operators that can change the model during the evaluation of a formula. In particular, we extend the basic modal language with modalities that are able to swap, delete or add pairs of related elements of the domain. We call the resulting logics *Relation-Changing Modal Logics*. We study local version of the operators (performing modifications from the evaluation point) and global version (changing arbitrarily edges in the model). We investigate several properties of the given languages, from an abstract point of view. First, we introduce the formal semantics of the model modifiers, afterwards we introduce a notion of bisimulation. Bisimulations are an important tool to investigate the expressive power of the languages introduced in this thesis. We show that all the languages are incomparable among them in terms of expressive power (except for the two versions of swap, which we conjecture are also incomparable). We continue by investigating the computational behaviour of this kind of operators. First, we prove that the satisfiability problem for some of the relation-changing modal logics we investigate is undecidable. Then, we prove that the model checking problem is PSpace-complete for the six logics. Finally, we investigate model checking fixing the model and fixing the formula (problems known as formula and program complexity, respectively). We show that it is possible to define complete but non-terminating methods to check satisfiability. We introduce tableau methods for relation-changing modal logics and we prove that all these methods are sound and complete, and we show some applications.

In the last part of the thesis, we discuss a concrete context in which we can apply relation-changing modal logics: *Dynamic Epistemic Logics* ($\mathcal{DEL}$). We motivate the use of the kind of logics that we investigate in this new framework, and we introduce some examples of $\mathcal{DEL}$. Finally, we define a new relation-changing modal logic that embeds $\mathcal{DEL}$ and we investigate its computational behaviour.

KEYWORDS: modal logics, relation-changing operators, dynamic operators, bisimulations, expressive power, complexity, decidability, dynamic epistemic logics.

Resumen

En esta tesis investigamos operadores modales dinámicos que pueden cambiar el modelo durante la evaluación de una fórmula. En particular, extendemos el lenguaje modal básico con modalidades que son capaces de invertir, borrar o agregar pares de elementos relacionados. Estudiamos la versión local de los operadores (es decir, la realización de modificaciones desde el punto de evaluación) y la versión global (cambiar arbitrariamente el modelo). Investigamos varias propiedades de los lenguajes introducidos, desde un punto de vista abstracto. En primer lugar, se introduce la semántica formal de los modificadores de modelo, e inmediatamente se introduce una noción de bisimulación. Las bisimulaciones son una herramienta importante para investigar el poder expresivo de los lenguajes introducidos en esta tesis. Se demostró que todas los lenguajes son incomparables entre sí en términos de poder expresivo (a excepción de los dos versiones de swap, aunque conjeturamos que también son incomparables). Continuamos por investigar el comportamiento computacional de este tipo de operadores. En primer lugar, demostramos que el problema de satisfactibilidad para las versiones locales de las lógicas que investigamos es indecidible. También demostramos que el problema de model checking es PSPACE-completo para las seis lógicas. Finalmente, investigamos model checking fijando el modelo y fijando la fórmula (problemas conocidos como complejidad de fórmula y complejidad del programa, respectivamente). Es posible también definir métodos para comprobar satisfactibilidad que no necesariamente terminan. Introducimos métodos de tableau para las lógicas que cambian las relaciones y demostramos que todos estos métodos son correctos y completos, y mostramos algunas aplicaciones.

En la última parte de la tesis, se discute un contexto concreto en el que pueden aplicarse las lógicas modales que cambian la relación: *Lógicas Dinámicas Epistémicas* ($\mathcal{DEL}$, por las siglas en inglés). Motivamos el uso del tipo de lógicas que investigamos en este nuevo marco, e introducimos algunos ejemplos de $\mathcal{DEL}$. Finalmente, definimos una lógica que cambia la relación capaz de codificar $\mathcal{DEL}$, e investigamos su comportamiento computacional.

PALABRAS CLAVE: lógicas modales, operadores de cambio de accesibilidad, operadores dinámicos, bisimulaciones, poder expresivo, complejidad, decidibilidad, lógicas dinámicas epistémicas.

Part I

WHY DYNAMIC MODAL LOGICS?

Mareos que dan gran jaleo
(Modales que son malas mañas).

from "Amnesia", Carlos 'Indio' Solari.

The first discussions about logic appeared in the 5th century *B.C.* in the Greek and Roman antiquity, with a particular interest in sentence analysis, truth and fallacies. Logic was used in its origins as an instrument to describe elements of the real world, and to answer the big questions of the universe. There are many evidences of the use of logic in philosophy, starting by Aristotle's syllogism, which represents the introduction of a formal system of thought, or the attempts of Kant and Descartes of using logic to demonstrate the existence of God, in more modern philosophy. These two examples, with a large difference of time (from the 4th century *B.C.* in the first case, to the 17th and 18th centuries *A.D.* in the last cases) are not isolated and show that philosophy has been the main partner of logic for a long time.

But in the 2nd century *A.D.* Galen already thought in logical constructions to help formalize mathematics. The link with mathematics became even more relevant with the work of Leibniz in the 17th century. From Leibniz through Boole, Frege, Cantor, Hilbert's attempt to axiomatize mathematics, until the incompleteness theorem of Gödel, logic and mathematics have lived together and for many years now these two disciplines have been closely tied.

Leibniz had also another amazing vision: he dreamed with machines capable to make calculations, to free humans of this task. In the 20th century, Turing and von Neumann made Leibniz's dream true. Turing formalized a logical machine capable of representing any computation, and later von Neumann designed the logical architecture for a concrete computation machine. These are the origins of *Computer Science*, and since then logic became both its foundations, and one of its most promising areas of application.

1 The Role of Logic in Computer Science

1.1 COMPUTATION AND LOGIC

Computer Science is a discipline which cannot be classified unambiguously as either an engineering, or a mathematical discipline. From the software development point of view, computer science is pure engineering: it covers the design, construction and evaluation of computer systems. In contrast, from a theoretical point of view, it is possible to understand and explain the fundamentals of this science mathematically. The field is characterized by this dichotomy: from a practical perspective, the goal is to solve problems with computer programs; but we would not be able to define what kind of problems are solvable, or how hard it is to solve a problem, if we do not study computer science (or, quoting E. W. Dijkstra, *computing science*[1]) from a mathematical perspective.

The design and implementation of a software project is nothing else that the development of a product. First, the users describe their requirements as clearly as possible to the developers. Then, developers make those requirements more specific, to design the structure of the software. After that, development starts, i.e., the process of writing and testing the code. Finally, the end product is delivered to the user. All these phases that we just described very simply, require to take many decisions: decisions of planning, how to specify the requirements of the user, which programming language is more appropriate, how to guarantee quality in the final product, and many others. It is exactly the kind of tasks that an *engineer* has to do to build a bridge, a car engine or a software: she/he has to design, build and maintain the product. This perspective of computer science is in general well known, and constitutes, perhaps, the mainstream; but in this thesis, we focus on computer science from a mathematical perspective. Taking into account the vast number of possible connections between mathematics and computation, we should be more specific: we will devote ourselves to *Computational Logic*.

Computational logic is the use of logic to reason about computation. There is a large number of applications of computational logic in different areas. Methods and concepts from logic have a lasting impact in computer science, with unexpected and effective results. In [Halpern *et al.*, 2001], concrete examples of such impact are introduced. Let us discuss in detail some of these examples to highlight how deep is the effect of logic within computer science.

Complexity Theory for instance, investigates the question *"How much time and how much memory space is needed to solve a particular problem?"*. This measure is done

[1] In his manuscripts, Dijkstra referred to "computing science" instead of "computer science", arguing that this science is about studying "how to compute", not computers.

by classifying a problem into complexity classes. Typical examples of complexity classes are polynomial time, non-deterministic polynomial time, polynomial space, exponential time, etc. It is possible to give descriptive characterizations of each of these classes as follows. If the fact that a problem can be defined by using the expressive power of a logic $\mathcal{L}$ ensures that the problem belongs to a complexity class $\mathcal{C}$, then $\mathcal{L}$ characterizes $\mathcal{C}$. This kind of characterizations are one of the goals of the field of Descriptive Complexity [Immerman, 1995]. Classical complexity classes such as those mentioned before have natural descriptive characterizations. For instance, Fagin's theorem [Fagin, 1974] characterizes the class NP with the existential fragment of Second-Order Logic. Thus, logic has been an effective tool for answering some of the basic questions in complexity theory.

Complexity theory is an example of the application of logic in theoretical computer science. But logic can also be useful in applied computer science. In practice, one of the main areas of computer science is the study of Databases [Maier, 1983], which is concerned with storing, querying and updating large amounts of data. Logics and databases are related since the early 1970s, given that there are logical languages than can also be seen as Database Query Languages. Indeed, many of the standard query systems such as SQL (Structured Query Language) or QBE (Query by Example) are based on first-order logic, and more powerful languages are extensions of first-order logic with recursion. The impact of logic on databases is one of the most remarkable examples of the effectiveness of logic in applied computer science.

Logic also helps define the fundamental properties of programming languages, probably one of the most important tools in the development of software products. Type Theory [Reynolds, 1985; Reynolds, 1998] is a formal framework for the design, analysis and implementation of programming languages. Thanks to type theory it is possible to describe complex concepts such as data abstraction, inheritance and polymorphism. This framework is based on logics of program behaviour that are appropriate to reason about programs. It let us present concepts modularly, and verify properties of programming languages.

Logic is present in other areas of computer science, in which certain kind of interaction between different objects is necessary, such as distributed computing, game theory and artificial intelligence. Classical questions that these areas need to answer are *"What do processes need to know about other processes in order to coordinate an action?" "What do agents need to know about other agents to carry on a conversation?" "What does a robot need to know in order to open a safe?"*. All these tasks require reasoning about knowledge and belief, a field initially motivated by the philosophy community in the 1950s [von Wright, 1951] and investigated later by Hintikka [Hintikka, 1962]. This initiates the study of epistemic logics, in which knowledge and interaction between agents are represented in models of *possible worlds*, providing a formal framework to represent the kind of computational systems that we mentioned before.

Finally, Software Verification is probably one of the most classical examples of computational logic, given that computation and logic are intrinsic parts of the problem. Computer programs and their behaviour are formalized in some logical language, and properties are verified on this representation. A particular case is the Model Checking task [Clarke, 2008]: given a property and one automaton representing a system, find all the states where the property holds. As we can see, we need some logical language to express properties on automatons, and the language has to posses some characteristics that makes the task *automatizable*.

We have listed several cases of computational problems solved with a logical approach. They are clear examples in which a logical approach directly lead to advances in computer science. In this thesis, we will investigate a family of logics that can be used to model dynamic environments. We define languages with a specific purpose (describe changes in relationships between elements), and we study their properties. For instance, we investigate what these languages can express, and what they cannot. We compare them to determine which is more appropriate for different reasoning tasks. We are also interested in their computational behaviour, for example, whether their inference tasks are decidable or not. In particular, we want to establish if the satisfiability problem is decidable or not, i.e., if given a formula we have an algorithm that answers *yes* if there is a model for the formula, and *no* otherwise. Another interesting question is the exact complexity of the following inference task: given a formula and a finite model, decide if the formula holds in such model. We are interested in the computational cost of performing those tasks, classifying them in complexity classes, such as P, NP, PSpace, etc. We also look for applications of the kind of languages that we investigate, for instance, to model epistemic scenarios, and we compare them with other languages already used in the area.

1.2 WHICH LOGIC?

We discussed the relation between logic and computation, but we did not say anything yet about *which logic* is best suited for the task. For many years, talk about logic was considered talk about *First-Order Logic* ($\mathcal{FOL}$) [Enderton, 1972]. This is not a surprise, given that many mathematicians included figures like Russell, Hilbert and Gödel, picked this logic for many years to answer the fundamental questions about mathematics. First-order logic is a language with well known properties, studied in detail for many years, which is easy to understand and very powerful.

To understand more about the behaviour of $\mathcal{FOL}$, let us introduce in detail the syntax and semantics of the First-Order Language and some of its properties:

Definition 1.2.1 (First-Order Language). *Let* $\mathsf{REL} = \{R_1, R_2, \ldots\}$ *be a countable set of* relation symbols, $\mathsf{FUN} = \{f_1, f_2, \ldots\}$ *a countable set of* function symbols, $\mathsf{CON} = \{c_1, c_2, \ldots\}$ *a countable set of* constant symbols *and* $\mathsf{VAR} = \{x_1, x_2, \ldots\}$ *a countable set of* variables. *We assume that* REL, FUN, CON *and* VAR *are pairwise disjoint. To each relation symbol* $R_i \in \mathsf{REL}$ *and each function symbol* $f_i \in \mathsf{FUN}$ *we associate an arity* $n > 0$. *We call* $\mathcal{S} = \langle \mathsf{REL}, \mathsf{FUN}, \mathsf{CON}, \mathsf{VAR} \rangle$ *a* signature, *and we will sometimes focus on* relational signatures *where* $\mathsf{FUN} = \{\}$. *This is usually not a restriction as we can represent functions as constrained relations.*

The well-formed terms *of the first-order language over the signature* $\langle \mathsf{REL}, \mathsf{FUN}, \mathsf{CON}, \mathsf{VAR} \rangle$ *are*

$$\mathsf{TERM} := x_i \mid c_i \mid f_i(t_1, \ldots, t_n),$$

where, $x_i \in \mathsf{VAR}$, $c_i \in \mathsf{CON}$, $f_i \in \mathsf{FUN}$ *of arity n and* $t_1, \ldots, t_n \in \mathsf{TERM}$. *The* well-formed formulas *over the signature are*

$$\mathsf{FORM} := \top \mid t_1 = t_2 \mid R_i(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x_i.\varphi,$$

where $t_1, t_2, \dots, t_n \in \mathsf{TERM}$, $R_i \in \mathsf{REL}$ *is an n-ary relation symbol,* $\varphi, \varphi_1, \varphi_2 \in \mathsf{FORM}$ *and* $x_i \in \mathsf{VAR}$*. As usual, we take* $\vee, \rightarrow, \leftrightarrow$ *and* $\forall$ *as defined symbols.*

Turning to semantics, first-order formulas are interpreted on first-order models.

Definition 1.2.2 (First-Order Models and Satisfiability)**.** *A first-order* model *for a signature* $\mathcal{S}$*, is a structure* $\mathcal{M} = \langle M, \cdot^{\mathcal{M}} \rangle$ *where M is a non-empty set and* $\cdot^{\mathcal{M}}$ *is an* interpretation function *defined over* $\mathsf{REL} \cup \mathsf{FUN} \cup \mathsf{CON}$ *such that* $\cdot^{\mathcal{M}}$ *assigns an n-ary relation over M to n-ary relation symbols in* REL*, an n-ary function* $^nM \to M$ *to n-ary function symbols in* FUN*, and an element in M to constant symbols in* CON*. When the signature is small, we will simple write* $\mathcal{M} = \langle M, \{R_i^{\mathcal{M}}\}, \{f_j^{\mathcal{M}}\}, \{c_k^{\mathcal{M}}\} \rangle$ *instead of* $\mathcal{M} = \langle M, \cdot^{\mathcal{M}} \rangle$*.*

An assignment *g for* $\mathcal{M}$ *is a mapping* $g : \mathsf{VAR} \to M$*. Given an assignment g for* $\mathcal{M}$*,* $x \in \mathsf{VAR}$ *and* $m \in M$*, we define* g^x_m *(an x-*variant *of g) by* $g^x_m(x) = m$ *and* $g^x_m(y) = g(y)$ *for* $x \neq y$*. Given a model* $\mathcal{M}$ *and an assignment g for* $\mathcal{M}$*, the interpretation function* $\cdot^{\mathcal{M}}$ *can be extended to all elements in* TERM*:*

$$\begin{array}{rcl} x_i^{\mathcal{M}} & = & g(x_i) \\ f(t_1, \dots, t_n)^{\mathcal{M}} & = & f^{\mathcal{M}}(t_1^{\mathcal{M}}, \dots, t_n^{\mathcal{M}}). \end{array}$$

Finally the satisfiability relation $\models$ *is defined as*

$$\begin{array}{rll} \mathcal{M} \models \top[g] & & \textit{always} \\ \mathcal{M} \models t_1 = t_2[g] & \textit{iff} & t_1^{\mathcal{M}} = t_2^{\mathcal{M}} \\ \mathcal{M} \models R(t_1, \dots, t_n)[g] & \textit{iff} & R^{\mathcal{M}}(t_1^{\mathcal{M}}, \dots, t_n^{\mathcal{M}}) \\ \mathcal{M} \models \neg\varphi[g] & \textit{iff} & \mathcal{M} \not\models \varphi[g] \\ \mathcal{M} \models \varphi_1 \wedge \varphi_2[g] & \textit{iff} & \mathcal{M} \models \varphi_1[g] \textit{ and } \mathcal{M} \models \varphi_2[g] \\ \mathcal{M} \models \exists x_i.\varphi[g] & \textit{iff} & \mathcal{M} \models \varphi[g^{x_i}_m] \textit{ for some } m \in M. \end{array}$$

If a given formula φ *is satisfied under every assignment for* $\mathcal{M}$*, we say that* φ *is* valid *in* $\mathcal{M}$ *and write* $\mathcal{M} \models \varphi$ *.*

$\mathcal{FOL}$ is a well understood and powerful language, but it also has important disadvantages. On the one hand, sometimes it is not sufficiently expressive. For example, it is well known that $\mathcal{FOL}$ cannot express the transitive closure of a relation, or the fact that a relation is well founded. On the other hand, and most importantly, from a computational perspective, $\mathcal{FOL}$ is sometimes, too expressive. The next result is a clear example of this situation:

Theorem 1.2.3. *The satisfiability problem for* $\mathcal{FOL}$ *is undecidable.*

Proof. There are several proofs for this theorem. In 1936 Church proved that no recursive function could decide the validity of first-order sentences, and concluded that there was no decision algorithm for the satisfiability problem of $\mathcal{FOL}$ [Church, 1936]. In 1937, Turing formalized Turing machines by means of first-order formulas, and reduced a particular class of undecidable problems for Turing machines to the validity problem [Turing, 1937]. In [Berger, 1966], we can find a proof by reducing the undecidable Tiling problem to $\mathcal{FOL}$ satisfiability. □

Theorem 1.2.3 establishes that there is no terminating procedure to check satisfiability of a first-order formula, which is one of the main problems for any logic. On the other hand, its model checking problem is decidable and PSPACE-complete [Stockmeyer, 1974; Chandra and Merlin, 1977; Vardi, 1982]. If we only need model checking, $\mathcal{FOL}$ can be a good choice, but in general if we need to model a problem with a logic having good computational properties, $\mathcal{FOL}$ or any more expressive logic might not be the best alternative. In this case, it is natural to look for new languages with better computational properties. Modal logics [Blackburn *et al.*, 2001; Blackburn and van Benthem, 2006] fall into this family of logics: they are logics designed to talk about relational structures, and that can be adapted according to the problem we are dealing with. Let us introduce the *Basic Modal Logic* $\mathcal{ML}$:

Definition 1.2.4 (Syntax of Basic Modal Logic). *Let* PROP *be a countable, infinite set of propositional symbols. Then the set* FORM *of formulas of* $\mathcal{ML}$ *over* PROP *is defined as:*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond\varphi,$$

where $p \in \mathsf{PROP}$ *and* $\varphi, \psi \in \mathsf{FORM}$. *Other operators are defined as usual:* $\top$ *is* $\neg\bot$, $\varphi \vee \psi$ *is defined as* $\neg(\neg\varphi \wedge \neg\psi)$ *and* $\Box\varphi$ *is a shorthand for* $\neg\Diamond\neg\varphi$.

Now we define the structures where we interpret formulas of the basic modal language. Models are just labeled directed graphs, which we call *Kripke Models*.

Definition 1.2.5 (Kripke Models). *A Kripke Model* $\mathcal{M}$ *is a triple* $\mathcal{M} = \langle W, R, V\rangle$, *where* W *is a non-empty set whose elements are called points or states;* $R \subseteq W \times W$ *is the accessibility relation; and* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *is a valuation. Let* w *be a state in* $\mathcal{M}$, *the pair* $(\mathcal{M}, w)$ *is a pointed model; we usually drop parentheses and call* $\mathcal{M}, w$ *a pointed model.*

Figure 1 shows an example of a Kripke model. As we can see, the model $\mathcal{M}$ is a graph with three elements, $\{w, v, u\}$. w is labeled by p, u by p and q, and v has no label. Formally, $\mathcal{M}{=}\langle W, R, V\rangle$, where $W{=}\{w, v, u\}$, $R{=}\{(w, v), (w, u), (v, v), (v, u), (u, v)\}$, and $V(p){=}\{w, u\}, V(q){=}\{u\}$.

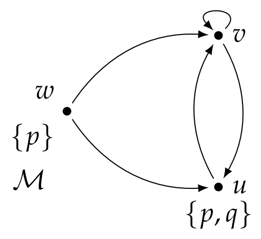


Figure 1: Example of a Kripke model.

We turn now to semantics. Operators in $\mathcal{ML}$ describe *local* properties of the models, which means that formulas are evaluated in some specific point. Pointed models are used for this.

Definition 1.2.6 (Semantics of Basic Modal Logic)**.** *Given a pointed model $\mathcal{M}, w$ and a formula φ we say that $\mathcal{M}, w$ satisfies φ (notation, $\mathcal{M}, w \models \varphi$) when*

$$\begin{array}{lll} \mathcal{M}, w \models \bot & & \textit{never} \\ \mathcal{M}, w \models p & \textit{iff} & w \in V(p) \\ \mathcal{M}, w \models \neg\varphi & \textit{iff} & \mathcal{M}, w \not\models \varphi \\ \mathcal{M}, w \models \varphi \wedge \psi & \textit{iff} & \mathcal{M}, w \models \varphi \textit{ and } \mathcal{M}, w \models \psi \\ \mathcal{M}, w \models \Diamond\varphi & \textit{iff} & \textit{for some } v \in W \textit{ s.t. } (w, v) \in R, \mathcal{M}, v \models \varphi. \end{array}$$

A formula φ of $\mathcal{ML}$ is satisfiable *if there exists a pointed model $\mathcal{M}, w$ such that $\mathcal{M}, w \models \varphi$.*

$\mathcal{ML}$ has better computational properties than $\mathcal{FOL}$. In fact, its satisfiability problem is decidable, and its complexity is PSPACE-complete. In addition, the model checking problem is in P. Let us introduce these results in detail. From Definition 1.2.4 it is clear that the language can be seen as an extension of propositional logic. But we can also show that $\mathcal{ML}$ is a fragment of first-order logic. In particular, it is a fragment of first-order logic with two variables ($\mathcal{FOL}^2$), which is decidable [Scott, 1962]. We can use this result to show the decidability of $\mathcal{ML}$. Notice that a modal model $\mathcal{M} = \langle W, R, V \rangle$ can be seen as a first-order model by considering V as the part of the interpretation function defining the meaning of unary predicate symbols. Semantics conditions in Definition 1.2.6 are purely first-order. Using these two intuitions, we can get a pair of mutually recursive functions, which translates formulas from $\mathcal{ML}$ to formulas in $\mathcal{FOL}^2$:

Definition 1.2.7 (Standard Translation into $\mathcal{FOL}^2$)**.** *Consider the signature $\mathcal{S} = \langle \{R\} \cup \{P_i \mid p_i \in \mathsf{PROP}\}, \{\}, \{\}, \{x, y\} \rangle$. The* standard translation *from $\mathcal{ML}$-formulas to $\mathcal{FOL}^2$-formulas over $\mathcal{S}$ is defined by two mutually recursive functions as:*

$$\begin{array}{rcl|rcl} ST_x(p_j) & = & P_j(x),\ p_j \in \mathsf{PROP} & ST_y(p_j) & = & P_j(y),\ p_j \in \mathsf{PROP} \\ ST_x(\neg\varphi) & = & \neg ST_x(\varphi) & ST_y(\neg\varphi) & = & \neg ST_y(\varphi) \\ ST_x(\varphi \wedge \psi) & = & ST_x(\varphi) \wedge ST_x(\psi) & ST_y(\varphi \wedge \psi) & = & ST_y(\varphi) \wedge ST_y(\psi) \\ ST_x(\Diamond\varphi) & = & \exists y.(R(x, y) \wedge ST_y(\varphi)) & ST_y(\Diamond\varphi) & = & \exists x.(R(y, x) \wedge ST_x(\varphi)) \end{array}$$

The next theorem establishes that the standard translation returns a formula which is equivalent to the original.

Theorem 1.2.8. *Let φ a $\mathcal{ML}$-formula, then for any model $\mathcal{M}$ we have*

$$\mathcal{M}, w \models \varphi \textit{ iff } \mathcal{M} \models ST_x(\varphi)\,[x \mapsto w]\,.$$

It has been proven that the satisfiability problem for $\mathcal{FOL}^2$ is decidable [Scott, 1962]. Actually [Grädel *et al.*, 1997] shows that the problem is NEXPTIME-complete. From these results we can conclude that the satisfiability problem for $\mathcal{ML}$ is also decidable and its upper bound is NEXPTIME.

Theorem 1.2.9. *Satisfiability problem for $\mathcal{ML}$ is decidable.*

Theorem 1.2.9 tells us that $\mathcal{ML}$ has better computational behaviour than $\mathcal{FOL}$ for the satisfiability problem. Let us now turn to the following model checking problem.

Given a finite pointed model $\mathcal{M}, w$ and a formula φ, determine if $\mathcal{M}, w \models \varphi$. Again, the basic modal logic $\mathcal{ML}$ behaves well for this task. For first-order logic, this problem is PSpace-complete but for $\mathcal{ML}$ it is in P. In [Blackburn and van Benthem, 2006] a polynomial bottom-up labeling algorithm solving the model checking problem for $\mathcal{ML}$ is introduced.

Theorem 1.2.10. *Model Checking for $\mathcal{ML}$ is in* P.

Proof. Given a formula φ, we label every point in the model with all the subformulas of φ that are true at this point. We label a point w with a propositional symbol p if and only if w is in the valuation of p. Boolean cases are handled in the obvious way. The only complex case is for modalities. We label w with $\Diamond\varphi$ if one of the successors is labeled by φ, and we label it with $\Box\varphi$ if all the successors are labeled by φ. The pseudo code of the procedure to check diamonds is displayed in Algorithm 1.

Algorithm 1 Model Checking $\Diamond\varphi$

```
procedure MC-$\Diamond(\psi)$
    $T \leftarrow \{v \mid \psi \in label(v)\}$
    while $T \neq \emptyset$ do
        choose $v \in T$
        $T \leftarrow T \setminus \{v\}$
        for all $w$ s.t. $R(w, v)$ do
            if $\Diamond\varphi \notin label(w)$ then
                $label(w) \leftarrow label(w) \cup \{\Diamond\psi\}$
            end if
        end for
    end while
end procedure
```

Once we label a point with φ we never duplicate work. The algorithm takes polynomial time in the size of the input and the model. The algorithm takes

$$con(\varphi) \times nodes(\mathcal{M}) \times nodes(\mathcal{M})$$

steps, where $con(\varphi)$ is the number of connectives of the formula, and $nodes(\mathcal{M})$ is the number of nodes in the model. $\Box$

We have seen that $\mathcal{ML}$ has good computational properties, but as we can see in Definition 1.2.7 it is a small fragment of $\mathcal{FOL}$. For instance, it is not possible to express in $\mathcal{ML}$ that a model has exactly three different elements, because this is not expressible in $\mathcal{FOL}^2$. We introduced $\mathcal{ML}$ as a particular case of modal logics, but in fact, there are many modal logics, each of them with its particular properties. In $\mathcal{ML}$, the semantics of $\Diamond$ and $\Box$ involves looking at the successors of the evaluation point. It means that we can always "move forward", but we can never "go back". We can consider the past operator, denoted by $\Diamond^{-1}$ with the following semantics:

$$\mathcal{M}, w \models \Diamond^{-1}\varphi \text{ iff } \text{ for some } v \in W \text{ s.t. } (v, w) \in R, \mathcal{M}, v \models \varphi.$$

Clearly, the semantics condition is a first-order formula. The translation ST of Definition 1.2.7 can be expanded to this new operator:

$$ST_x(\Diamond^{-1}\varphi) \quad = \quad \exists y.(R(y,x) \wedge ST_y(\varphi)) \;\big|\; ST_y(\Diamond^{-1}\varphi) \quad = \quad \exists x.(R(x,y) \wedge ST_x(\varphi))$$

Again, $\mathcal{ML}$ with the past operator is a fragment of $\mathcal{FOL}^2$, therefore it is decidable and its complexity upper bound is NExpTime.

Both $\Diamond$ and $\Diamond^{-1}$ depend on the accessibility relation: they are called *local* operators. It is also possible to add operations to describe unreachable parts of the model, i.e., to express *global* properties. The universal modalities were designed for this purpose. The E operator is an existential quantifier over all the states of the model. Its dual, the A operator, represents universal quantification. Their formal semantics is the following:

$$\begin{array}{lll} \mathcal{M},w \models \mathsf{E}\varphi & \text{iff} & \text{for some } v \in W, \mathcal{M},v \models \varphi \\ \mathcal{M},w \models \mathsf{A}\varphi & \text{iff} & \text{for all } v \in W, \mathcal{M},v \models \varphi. \end{array}$$

The standard translation can be also extended for the universal modality:

$$\begin{array}{lll|lll} ST_x(\mathsf{E}\varphi) & = & \exists y.(ST_y(\varphi)) & ST_y(\mathsf{E}\varphi) & = & \exists x.(ST_x(\varphi)) \\ ST_x(\mathsf{A}\varphi) & = & \forall y.(ST_y(\varphi)) & ST_y(\mathsf{A}\varphi) & = & \forall x.(ST_x(\varphi)) \end{array}$$

Clearly, the universal modalities increase the expressive power of the language (for instance, it can talk about isolated parts of a model), but it is still a proper fragment of $\mathcal{FOL}^2$. Hence, the $\mathcal{ML}$ equipped with E has a NExpTime upper bound. However, it has a more complex computational behaviour than the other examples: satisfiability for $\mathcal{ML}$ + E is ExpTime-complete [Spaan, 1993; Hemaspaandra, 1996].

We can see that modal logics are fragments of $\mathcal{FOL}$ with good computational properties. In some sense, they have a *dynamic* behaviour. Each time that we evaluate, for instance, a $\Diamond$, $\Diamond^{-1}$ or E we are moving to a different state, i.e., to a different pointed model. But, their dynamic power seems limited. Other operators were specially introduced to characterize dynamic situations. One of the most classical examples is Propositional Dynamic Logic, or **PDL** [Ladner, 1977; Fischer and Ladner, 1979; Harel, 1984]. This logic was created to represent executions of programs, modeling the changes of a state after the application of an action. **PDL** is dynamic in the sense of representing *behaviour*, however it never transforms the model. Other dynamic modal logics directly modify the model while evaluating a formula. For instance, Public Announcement Logic ($\mathcal{PAL}$) [Plaza, 2007; van Ditmarsch *et al.*, 2007] is a logic with an operator which deletes those states of the model which do not verify some property. Clearly, evaluating these operators we get a new model, in which some of the original states have been removed. This is the kind of languages that interests us in this thesis. More precisely, we investigate operations which change the accessibility relation in relational structures.

2

Introducing Dynamic Logics (by Example)

2.1 FROM BASIC MODAL LOGIC TO THE DYNAMIC APPROACH

Modal logics were born originally as a formalism to talk about *modes of truth*. Their origins can be attributed to C. I. Lewis [Lewis, 1918], who tried to solve paradoxes of the implication operator using the concepts of *necessity* and *possibility*. But this was just the beginning. For many years, modal logics have evolved and became a powerful tool in different areas. They have been used successfully in mathematics and computer science, in (computational) linguistics, economics, artificial intelligence, knowledge representation, philosophy, game theory, etc. Modal logics are particularly interesting because many problems can be modeled using *relational structures*, i.e. *graphs*, and modal languages are particularly appropriate to describe labeled directed graphs.

Necessity and possibility are examples of modes of truth that we can describe with modal languages. Other natural examples can be found in the literature. For instance, in Temporal Logics [Prior, 1957] we can capture the modes *"eventually in the future"* or *"always since some point in the future"*. Another classical example are Epistemic Logics [von Wright, 1951], which formalize the concepts of *knowledge* and *belief*. But, under all these interpretations we are describing properties of the graphs from a static point of view. Using these modal operators we can navigate and explore properties of a given structure, but we cannot change the structure itself. If we check a property of a node, apply some operations and then check again the same property on the same node, the property is not modified. After any number of applications of the modal operations, the structure remains unchanged.

Now, a question that we can postulate is:

> *What happens if we need to model situations in which the graph can change after the application of some operations?*

A first option could be modeling all the possible scenarios in different graphs obtained by changing the original one, and then use classical modal logics to describe them. Another option (which will be our approach) is to capture the *dynamic* behaviour internally in the language. We should take some care here, because some modal operators have been devised in the past to model dynamic phenomena, but not in the sense we just mentioned.

One example which we already mentioned in the previous chapter is *Propositional Dynamic Logic* (**PDL**). This logic is a formal system for reasoning about programs. Originally, it was designed to formalize correctness specifications and prove that those specifications correspond to a particular program. **PDL** is a modal logic that contains an infinite number of modalities $\langle\pi\rangle$, where each π corresponds to a *program*. The interpretation of $\langle\pi\rangle\varphi$ is that *"some terminating execution of π from the current state leads to a state where the property φ holds"*. The structure of a program is defined inductively from a set of basic programs $\{a, b, c, \ldots\}$ as:

- **Choice:** if π and π' are programs, then $\pi \cup \pi'$ is a program which executes non-deterministically π or π'.
- **Composition:** if π and π' are programs, then $\pi;\pi'$ is a program which executes first π and then π'.
- **Iteration:** if π is a program, π^* is the program that executes a finite number (possibly zero) of times π.
- **Test:** if φ is a formula, then $\varphi?$ is a program. It tests whether φ holds, and if so, continues; if not, it fails.

The semantics of **PDL**-formulas is straightforward: diamonds quantify existentially over the edges of a model, choosing non-deterministically or composing edges, iterating on the edges, or testing properties. Formally:

$$\mathcal{M}, w \models \langle\pi\rangle\varphi \text{ iff for some } v \text{ s.t. } (w,v) \in R_\pi, \mathcal{M}, v \models \varphi$$

where R_π either is the accessibility relation R_a corresponding to an atomic program a, or is defined inductively as:

$$\begin{array}{lcl} R_{\pi\cup\pi} & = & R_\pi \cup R_{\pi'} \\ R_{\pi;\pi} & = & R_\pi \circ R_{\pi'} \\ R_{\pi^*} & = & (R_\pi)^* \\ R_{\psi?} & = & \{(w,w) \mid \mathcal{M}, w \models \psi\}. \end{array}$$

The expressive power of **PDL** is high (notice that it goes beyond first-order logic, as it can express the reflexive-transitive closure of a relation), and **PDL** can express some interesting properties. For example the formula

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

represents that the program "**while** φ **do** a" ends in a state satisfying ψ. The program inside the modality executes a a finite, but not specified number of times after checking that φ holds, and after finishing the loop $\neg\varphi$ must holds. This captures exactly the behaviour of a while loop.

Clearly, the language gives us a practical way to deal with the notion of state and change, but this is a weak notion of dynamic behaviour. Formulas do not change the model, they only formalize program executions. The purpose of this thesis is to investigate operations that can change the model while we are evaluating a formula. We will see in the next section, various concrete examples of this kind of logics.

2.2 (REALLY) DYNAMIC MODAL LOGICS

2.2.1 *Hybrid Logics*

We have seen that modal models are labeled directed graphs. Propositional symbols act as labels or decorations on the nodes, and modal formulas are able to consult the properties of those labels and access adjacent nodes. Consider now operators that can change the model. One possibility could be to define an operator that *"re–decorates"* nodes in the model. Consider a new operator, that changes the label of the evaluation point. This is the effect of $\downarrow$ in the *Hybrid Logic* $\mathcal{HL}(@, \downarrow)$ [Blackburn and Seligman, 1995; Areces *et al.*, 2001; ten Cate, 2005; Areces and ten Cate, 2006]. The syntax and semantics of this language are extensions of the basic modal logic $\mathcal{ML}$. Hybrid logics involve a special set of propositional symbols called *nominals*, that act as names pointing to a unique state in the model.

$\mathcal{HL}(@, \downarrow)$ introduces the operators @ and $\downarrow$. @ is called the *satisfaction operator* and the formula $@_n\varphi$ states that φ is true at the unique state where n holds. The *down-arrow binder* $\downarrow$ binds a given nominal to the current state in the model. Hence, $\downarrow n.\varphi$ intuitively means "after naming the current state n, φ holds". $\mathcal{HL}(@, \downarrow)$ is more expressive than $\mathcal{ML}$. In fact, it is a reduction class of first-order logic [Areces *et al.*, 1999; ten Cate, 2005; Areces and ten Cate, 2006]. Let us introduce it formally.

Definition 2.2.1 (Syntax of Hybrid Logics). *Let the signature* $\langle \mathsf{PROP}, \mathsf{NOM} \rangle$ *be given, with* $\mathsf{NOM} \subseteq \mathsf{PROP}$. *The set* FORM *of formulas of* $\mathcal{HL}(@, \downarrow)$ *over* $\langle \mathsf{PROP}, \mathsf{NOM} \rangle$ *is defined as:*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond\varphi \mid @_n\varphi \mid \downarrow n.\varphi,$$

where $p \in \mathsf{PROP}$, $n \in \mathsf{NOM}$ *and* $\varphi, \psi \in \mathsf{FORM}$.

Syntactically, nominals can appear at the same places as regular propositional symbols, but they can also appear as parameters of the operators @ and $\downarrow$. Models of hybrid logics are similar to models of $\mathcal{ML}$. However they make each nominal point to exactly one state.

Definition 2.2.2 (Semantics of Hybrid Logics). *A hybrid model* $\mathcal{M}$ *is a triple* $\langle W, R, V \rangle$ *where* W *is non empty,* $R \subseteq W \times W$ *and* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *is a valuation such that* $V(n)$ *is a singleton if* $n \in \mathsf{NOM}$. *Let* $\mathcal{M}$ *be a hybrid model,* w *a state in* $\mathcal{M}$, *the semantics of the hybrid operators is defined as:*

$$\begin{array}{lll} \langle W, R, V \rangle, w \models @_n\varphi & \textit{iff} & \langle W, R, V \rangle, v \models \varphi \textit{ for } V(n) = \{v\} \\ \langle W, R, V \rangle, w \models \downarrow n.\varphi & \textit{iff} & \langle W, R, V_n^w \rangle, w \models \varphi, \end{array}$$

where V_n^w *is defined as* $V_n^w(n) = \{w\}$ *and* $V_n^w(m) = V(m)$, *for* $m \neq n$.

φ is satisfiable if for some pointed model $\mathcal{M}, w$ we have $\mathcal{M}, w \models \varphi$.

Let us discuss an example to show the expressive power of hybrid operators.

Example 2.2.3. *The formula* $\downarrow n.\Diamond n$ *states* "call the actual point n and ensure that n is reachable in one step", *which forces the current point to be reflexive.*

As we can see, $\mathcal{HL}(@, \downarrow)$ is a very powerful language, and the prize to pay is its high computational complexity. The satisfiability problem of $\mathcal{HL}(@, \downarrow)$ is undecidable [Areces *et al.*, 1999] and model checking is PSPACE-complete [Franceschet and de Rijke, 2003].

2.2.2 *Memory Logics*

Other dynamic languages have been studied, looking for better computational properties that those of $\mathcal{HL}(@, \downarrow)$. *Memory Logics* [Mera, 2009] are modal logics with the ability to *store* the current state of evaluation into a set (the memory) and to consult whether the current state of evaluation belongs to this set. Storing elements is equivalent to labeling a node as "visited" and later on, we will be able to check if the current point of evaluation has been visited before. Clearly memory languages are weaker than hybrid languages, because even though we can remember which states have been visited, we cannot know which of them we are currently visiting. There are several memory logics, we will just introduce two of them formally.

Definition 2.2.4 (Syntax of Memory Logics)**.** *Given a set* PROP*, the set* FORM *of formulas of $\mathcal{ML}(ⓡ, ⓚ)$ over* PROP *is defined as:*

$$\mathsf{FORM} ::= \bot \mid p \mid ⓚ \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond\varphi \mid ⓡ\varphi,$$

where $p \in$ PROP and $\varphi, \psi \in$ FORM.

Given a set PROP*, the set* FORM *of formulas of $\mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$ over* PROP *is defined as:*

$$\mathsf{FORM} ::= \bot \mid p \mid ⓚ \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle\!\langle \mathrm{r} \rangle\!\rangle\varphi,$$

where $p \in$ PROP and $\varphi, \psi \in$ FORM.

We turn now to semantics. Models of memory logics are modal models, but with an extra set where we store the elements that we visited.

Definition 2.2.5 (Semantics of Memory Logics)**.** *A model $\mathcal{M} = \langle W, R, V, S \rangle$ is an extension of an Kripke model with a memory $S \subseteq W$. Let w be a state in $\mathcal{M}$, we inductively define the notion of satisfiability of a formula as:*

$$\begin{array}{lll}
\langle W,R,V,S\rangle, w \models ⓚ & \textit{iff} & w \in S \\
\langle W,R,V,S\rangle, w \models ⓡ\varphi & \textit{iff} & \langle W,R,V,S \cup \{w\}\rangle, w \models \varphi \\
\langle W,R,V,S\rangle, w \models \langle\!\langle \mathrm{r} \rangle\!\rangle\varphi & \textit{iff} & \langle W,R,V,S\rangle, w \models ⓡ\Diamond\varphi.
\end{array}$$

A formula φ of $\mathcal{ML}(ⓡ, ⓚ)$ or $\mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$ is satisfiable *if there exists a model $\langle W, R, V, \emptyset \rangle$ such that $\langle W, R, V, \emptyset \rangle, w \models \varphi$.*

In the definition of satisfaction, the empty initial memory ensures that no point of the model satisfies the unary predicate ⓚ unless a formula $ⓡ\varphi$ or $\langle\!\langle \mathrm{r} \rangle\!\rangle\varphi$ has previously been evaluated there. The memory logic $\mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$ does not have the $\Diamond$ operator, and its expressive power is strictly weaker than $\mathcal{ML}(ⓡ, ⓚ)$ [Mera, 2009; Areces *et al.*, 2011]. However, in both cases we have a logic that is strictly more expressive than the basic modal logic $\mathcal{ML}$. We show this result with a simple example.

Example 2.2.6. *Given a pointed model $\langle W, R, V, \emptyset \rangle, w$, the $\mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$-formula $\langle\!\langle \mathrm{r} \rangle\!\rangle ⓚ$ is satisfiable only if w is reflexive. The $\langle\!\langle \mathrm{r} \rangle\!\rangle$ operator remembers the current element but at the same time looks for a successor. In this case, such a successor has to be in the memory, but w is the only one belonging to the memory (remember that we started with the empty memory). Then, the formula is satisfiable if only if w is his own successor. The same effect can be captured with the $\mathcal{ML}(ⓡ, ⓚ)$-formula $ⓡ\Diamond ⓚ$.*

As we mentioned before, memory logics are strictly weaker than hybrid logics. This is a hopeful result looking for good computational properties. The satisfiability problem for $\mathcal{ML}(\langle\langle r\rangle\rangle, ⓚ)$ is decidable but the one of $\mathcal{ML}(ⓡ, ⓚ)$ is not and its model checking problem is PSPACE-complete [Areces *et al.*, 2008; Areces *et al.*, 2009; Mera, 2009].

The two examples of dynamic logics we presented let us modify the labeling of nodes: hybrid logics allow to rename the elements of the model and memory logics mark nodes as already visited. The next examples will introduce logics which can modify the accessibility relation.

2.2.3 *Arrow Updates*

Arrow Update Logic (AUL) [Kooi and Renne, 2011a] was defined with the goal of modeling epistemic scenarios. This is presented as a theory of epistemic access elimination that generalizes previous works in the field. In [Kooi and Renne, 2011a], epistemic arrows are eliminated, no new arrows are created. Let us present formally the language.

Definition 2.2.7 (Syntax of Arrow Update Logic)**.** *Given a countable inifite set* PROP *of propositional symbols, and a finite set* AGT *of agent symbols, the set* FORM *of formulas of* AUL *over* PROP *and* AGT *is defined as:*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond_a\varphi \mid \langle U\rangle\varphi,$$

where $p \in \mathsf{PROP}$, $a \in \mathsf{AGT}$, $\varphi, \psi \in \mathsf{FORM}$ *and* $U \in \mathsf{UPD}$.

We need to define now the set UPD *of updates:*

$$\mathsf{UPD} ::= (\varphi, a, \psi) \mid (\varphi, a, \psi), U,$$

where $a \in \mathsf{AGT}$, $\varphi, \psi \in \mathsf{FORM}$ *and* $U \in \mathsf{UPD}$.

As we mentioned before AUL was defined to modeling epistemic scenarios. For this reason it was introduced in a multimodal or *multiagent* framework. The epistemic interpretation of an arrow labeled by a between the points w and v, is that the agent a cannot distinguish between the information contained in points w and v. We assume that each relation is reflexive, symmetric and transitive (it is an equivalence relation).

Definition 2.2.8 (Multiagent Kripke Models)**.** *A multimodal model* $\mathcal{M}$ *is a triple* $\mathcal{M} = \langle W, R, V\rangle$, *where* W *is a non-empty set whose elements are called points or states; for each* $a \in \mathsf{AGT}$, $R(a) \subseteq W \times W$ *is an accessibility relation (we will often write* R_a *rather than* $R(a)$*); and* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *is a valuation.*

Turning to semantics, diamonds represent the knowledge of an agent ($\Diamond_a\varphi$ is interpreted as "agent a considers that it is possible that φ"). The semantics of $\Diamond_a$ is the same we introduced in Definition 1.2.6, but each $\Diamond_a$ has associated a different R_a. The interest case is the update operator, which modifies the knowledge of the agent.

Definition 2.2.9 (Semantics of Arrow Update Logic)**.** *Given a multimodal model* $\mathcal{M} = \langle W, R, V\rangle$ *and* w *be a state in* $\mathcal{M}$, *we define the semantics of the update operator as:*

$$\begin{array}{lcl} \mathcal{M}, w \models \langle U\rangle\varphi & \textit{iff} & (\mathcal{M} * U), w \models \varphi \\ (\mathcal{M} * U) & = & \langle W, R', V\rangle \\ R'_a & = & \{(v, v') \in R_a \mid \exists(\varphi, a, \varphi') \in U : \mathcal{M}, v \models \varphi \textit{ and } \mathcal{M}, v' \models \varphi'\}. \end{array}$$

A formula φ of AUL *is* satisfiable *if there exists a model $\mathcal{M}$ such that $\mathcal{M}, w \models \varphi$.*

An update U characterizes the set of edges we want to keep in the accessibility relation. Each $(\varphi, a, \psi) \in U$ characterizes a-edges with a source point satisfying φ and a target point satisfying ψ. Let us see an example.

Example 2.2.10. *The update $\langle(\top, a, \Diamond_a \top)\rangle$ deletes all the a-edges such that the target point has not successors. After the update we get a model in which we can find dead ends one step earlier than in the original model.*

AUL has been introduced as a multi-agent language to reasoning about belief changes, by doing arrow eliminations or arrow updates. In [Kooi and Renne, 2011a] a sound and complete axiomatization for this logics is given, but its computational behaviour is not investigated in detail. AUL is capable of encoding Public Announcements [Plaza, 2007; van Ditmarsch *et al.*, 2007], the most "classical" logic used in the epistemic field.

2.2.4 *Sabotage Logic*

Another clear example of model-changing logics is *Sabotage Logic* introduced by Johan van Benthem in [van Benthem, 2005]. Consider the following *sabotage game*. It is played on a graph by two players, Runner and Blocker. Runner can move on the graph from node to accessible node, starting from a designated point, and with the goal of reaching a given final point. He should move one edge at a time. Blocker, on the other hand, can delete one edge from the graph, every time it is his turn. Runner wins if he manage to move from the origin to the final point in the graph, while Blocker wins otherwise. van Benthem proposes transforming the sabotage game into a modal logic, by working on models where edges are treated as objects and introducing the following "cross-model modality" referring to submodels from which objects have been removed:

Definition 2.2.11 (Syntax of Sabotage Modal Logic)**.** *Given a set* PROP*, the set* FORM *of formulas of SML over* PROP *is defined as:*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond\varphi \mid \blacklozenge\varphi,$$

where $p \in$ PROP and $\varphi, \psi \in$ FORM.

Definition 2.2.12 (Semantics of Sabotage Modal Logic)**.** *Given a model $\mathcal{M} = \langle W, R, V\rangle$ and w be a state in $\mathcal{M}$, we inductively define the notion of satisfiability of a formula as:*

$$\mathcal{M}, w \models \blacklozenge\varphi \quad \textit{iff} \quad \textit{there is } v \in W \textit{ s.t } v \neq w \wedge \mathcal{M} \setminus \{v\}, w \models \varphi.$$

A formula φ of SML is satisfiable *if there exists a model $\mathcal{M}$ such that $\mathcal{M}, w \models \varphi$.*

As a modal logic, it is clear that the $\blacklozenge$ operator *changes* the model in which a formula is evaluated. As van Benthem puts it, $\blacklozenge$ is an "external" modality that takes evaluation to another model, obtained from the current one by deleting some state or transition. Let see an example of the use of $\blacklozenge$ extracted from [van Benthem, 2005].

Example 2.2.13. *Consider the formula* ⟐$\Box\bot$. *It is satisfiable in an irreflexive 2-cycle, but it fails in a model consisting of a single reflexive point (the basic modal language cannot distinguish between these two models). In both cases we delete the only successor of the evaluation point, but after deletion in the reflexive model we get an empty model.*

Various sabotage modal logics have been studied [Löding and Rohde, 2003a; Löding and Rohde, 2003b; Rohde, 2006]. In particular, it has been proved that solving the sabotage game is PSpace-hard, while the model checking problem of the associated modal logic is PSpace-complete and the satisfiability problem for a close variant of this logic is undecidable. The logic fails to have both the finite model property and the tree model property. A translation of the sabotage modal logic into first-order logic is also provided.

2.2.5 *And More...*

There are many other examples of logics with operators that change the structure of the model. In [Aucher *et al.*, 2009], two logics to reason about modifications of graphs are introduced, named *Global and Local Graph Modifiers*. They include an operator $[\alpha]$, which is a box-like operator where α is a modality that can modify the model. For instance, given a model $\mathcal{M} = \langle W, R, V \rangle$, the formula

$$[p{-}\varphi]\psi$$

stipulates that all the states which satisfy φ are removed from $V(p)$, and we evaluate ψ in the model with the updated valuation. This modality changes the valuation of a propositional symbol. These languages include also primitives to change the domain and the accessibility relation. If we write

$$[\mathsf{nw}]\psi$$

we add a new point to the model, and the evaluation of ψ continues in the model with the new point as the evaluation point. The formula

$$[a{+}(\varphi, \varphi')]\psi$$

changes the accessibility relation: ψ is evaluated in a model that results to add a-edges from φ points to φ' points in $\mathcal{M}$.

The logics of graph modifiers include the operators to add and delete states and edges of the model. An extension including local modifiers is also investigated. These operators only add or remove the evaluation point in the valuation of some propositional symbol. In [Aucher *et al.*, 2009] graph modifiers are related with epistemic logics, showing that the logic with global modifiers generalizes public announcements. The article also shows that when local modifiers are included, the resulting logic is undecidable. In [Pucella and Weissman, 2004] these logics have been studied in a deontic framework, where adding edges represents granted permissions and deleting edges corresponds to revoking permissions.

Dynamic logics are also used in the field of Natural Language Processing (NLP). In [Groenendijk and Stokhof, 1991], a system using *Dynamic Predicate Logic* (DPL) is introduced to obtain a theory of discourse semantics. The language of DPL is composed by n-place predicates, constants and variables. They are interpreted as we introduced in Chapter 1 for $\mathcal{FOL}$. The models used are ordinary first-order models, consisting of a domain M of individuals and an interpretation function $\cdot^{\mathcal{M}}$, assigning individuals to the constants, and sets of n-tuples of individuals to the n-place predicates. Assignments are used as usual, i.e., as total functions from the set of variables to the domain. In the dynamic semantics of DPL, the semantic object expressed by a formula is a set of ordered pairs of assignments $\langle h, g\rangle$, that can be treated as an input-output pair. For instance, the dynamic interpretation of $\exists x.\phi$ consists of those pairs of assignments $\langle g, h\rangle$ such that there is some assignment k which differs from g at most in x and which together with h forms a possible input-output pair for ϕ. Also, a dynamic conjunction is defined. $\phi \wedge \psi$ with input g may result in output h if there is some k such that interpreting ϕ in g may lead to k, and interpreting ψ in k enables us to reach h.

This idea of defining dynamically the semantics of the operators, gives the power of getting a compositional treatment of sentences, for instance, when some part of the conjunction depends on the other. With DPL, we can represent the sentence "A man walks in the park. He whistles." as

$$\exists x.[\texttt{walk\_in\_the\_park}(x)] \wedge \texttt{whistle}(x)$$

because the occurrence of x in the predicate whistle is interpreted as in the output of the existential formula. What is interesting for us is the dynamic component in DPL, which is that the values of the assignment functions can change after evaluating a formula.

As a last example we can mention *Real-time Logics*, which were introduced to model real-time systems. In [Alur and Henzinger, 1994; Demri *et al.*, 2007], temporal logics are enriched with the operator $x.$ called *the freeze quantifier*. This is a quantification in which x is bound to the time of a particular state. We can explain the behaviour of the new quantifier with an example. For instance, the typical time-bounded response requirement that every request p is followed by a response q within 10 time units, can be expressed by the formula

$$\Box x.(p \rightarrow \Diamond y.(q \wedge y \leq x + 10)).$$

It can be read as "whenever there is a request p, and the variable x is frozen to the current time, the request is followed by an answer q at the time y, such that y is at most 10 more units than x". Is has been proved in [Alur and Henzinger, 1994; Demri *et al.*, 2007] that the satisfiability problem for logics augmented with the freeze quantifier, and interpreted over timed state sequences (sequences of states, each of which is labeled with a time from a discrete time domain) is ExpSpace-complete.

In this thesis we are interested in some questions about the general behaviour of model modifiers. More precisely, we are interested in operators which change the accessibility relation of a model. There are many situations in which this kind of operators is used. Operators that change the accessibility relation are appropriate to model scenarios, such as changes in the knowledge of an agent in epistemic logics.

2.3 THIS THESIS

The discussion in previous sections gives an intuitive idea of the topic of this thesis. We aim to study modal operators that allow to change the structure of a graph. In particular, we are interested in modal logics that modify the accessibility relation of a model. Part II is where we introduce several relation-changing logics with local and global effects. We discuss three modifiers: *Sabotage*, which deletes edges of the model; *Swap*, that turns around edges; and *Bridge*, which adds new edges to the model. All of them can work locally, i.e. modifying adjacent edges from the evaluation point, or globally, changing edges anywhere in the model. In Chapter 3 we motivate the use of relation-changing logics, and we introduce the primitives. We then give an example that shows some complex behaviour of this kind of logics: the lack of the tree and finite model property. Most of these results were previously presented in [Areces *et al.*, 2012; Fervari, 2012; Areces *et al.*, 2013b].

In Chapter 4 we present a new definition of bisimulation that is appropriate for the logics we are investigating, and we show that this definition captures exactly the intended meaning. Then, we show that all the logics introduced are incomparable in expressive power (comparison between local and global swap is open, but we conjecture they are also incomparable). Chapter 5 investigates the computational properties of these logics. We study the *Satisfiability Problem* in detail, and we prove that, at least for the local cases, it is undecidable. This is the starting point for the following chapter, in which we look for other computational tasks that behave better than satisfiability, such as *Model Checking*. Based on results from [Areces *et al.*, 2012] we prove in Chapter 6 that model checking for all these logics is decidable and PSPACE-complete. At the end of the chapter we also study two other problems: *program complexity* and *formula complexity*. Then, in Chapter 7 we introduce *Tableau Methods* as we define in [Areces *et al.*, 2013c] for all the logics. Later we investigate two possible applications for tableau methods: tableaux as a model building procedure, which combined with model checking gives us a terminate-and-check procedure for the satisfiability problem, and we discuss how tableaux can be used as a constructive method to compute interpolants.

In Part III we show applications of the logics studied in this thesis. First, we introduce Dynamic Epistemic Logics and then we show how they can be translated to a relation-changing logic. We study some properties of this new logic, and we explain the advantages and disadvantages of working in this framework.

Finally, in Part IV we discuss some conclusions and remarks, and we point to directions of future research.

Part II

ABSTRACT RELATION-CHANGING MODAL LOGICS

Insisto sobre el carácter inventivo que hay en cualquier lenguaje, y lo hago con intención. La lengua es edificadora de realidades. Las diversas disciplinas de la inteligencia han agenciado mundos propios y poseen un vocabulario privativo para detallarlos. Las matemáticas manejan su lenguaje especial hecho de guarismos y signos y no inferior en sutileza a ninguno.

from "Palabrería para versos", Jorge Luis Borges.

In this part of the thesis we will introduce a family of logics defined to represent dynamic behaviour. In particular, we will investigate logics that can modify the accessibility relation of a model while we evaluate a formula. We call this family *Relation-Changing Modal Logics*.

We will introduce six relation-changing operators: *swap*, turns around edges of the model; *sabotage* (introduced before by van Benthem [van Benthem, 2005]) deletes edges; and *bridge*, which adds new edges in the model to unaccessible states. All of them are introduced in two versions: *local*, i.e., changing edges from the evaluation point, and *global*, i.e., modifying arbitrary edges in the model.

Several relation-changing logics were investigated in previous works. As we mentioned, van Benthem introduced sabotage logic to formalize sabotage games. More technical results about sabotage logic were investigated in [Löding and Rohde, 2003b; Rohde, 2006]. In [Aucher *et al.*, 2009] some relation-changing operators have been investigated as data structure modifiers. In [Kooi and Renne, 2011a; Kooi and Renne, 2011b], operators which delete edges depending on their pre and postcondition were introduced. Our goal, is to study relation-changing modal logics from an abstract point of view, and investigate in detail the consequences of including these kind of operators in modal logics.

We will start in Chapter 3 by introducing the primitives and showing some model properties. In Chapter 4 we will introduce the tools to investigate the expressive power of the languages: *bisimulations* and *Ehrenfeucht-Fraïssé Games*. We will use them to establish what is the relation among the six relation-changing modal logics we defined, according their expressive power. We will continue by investigating the computational behaviour of the logics. Chapter 5 is devoted to investigate *the satisfiability problem* of relation-changing modal logics, and Chapter 6 is dedicated to *the model checking* problem. Finally, we will close this part by introducing *tableau methods* for these logics.

In this part of the thesis, we will discuss the gains and losses of using relation-changing operators. We will see that even though it is natural to use these primitives to model dynamic scenarios, there is a trade-off between their expressive power and their computational complexity.

3

Relation-Changing Modal Logics

Ya no me digas que se siente. Si no se cambia hoy, no se cambia más...

from "Agua de la Miseria", Luis Alberto Spinetta.

3.1 CHANGING THE ACCESS

Many situations in real life are dynamic. The way in which certain scenario changes after determined action, different external factors modifying the environment or simply the impact of time, in a very general way, are *dynamic*. For instance, every day, the people living at Córdoba City get up in the morning assuming that a bus will stop at the same place as yesterday. But, reading the morning news, they find out that there will be no buses running in the city for a couple of days as a result of a strike. After two days, as promised, service returns to normality. In this situation, some factor (the strike) *changed* the usual scenario, i.e., getting up, going to the stop to get the bus and arriving at work. After certain time things changed back to normality.

Let us consider now a more interesting scenario where the strike is partial, only some of the bus lines were affected. This is an instance of the scenario proposed by Johan van Benthem in [van Benthem, 2005], in which some "malevolent" factor (in this case the strike) makes the connections from home to work disappear. As van Benthem proposed, this situation can be modeled as a two player game we can call a *sabotage game*. We can represent the bus lines as a graph, where two points are connected if there is a bus linking one to the other. Dotted lines represent paths where buses were interrupted by the strike.

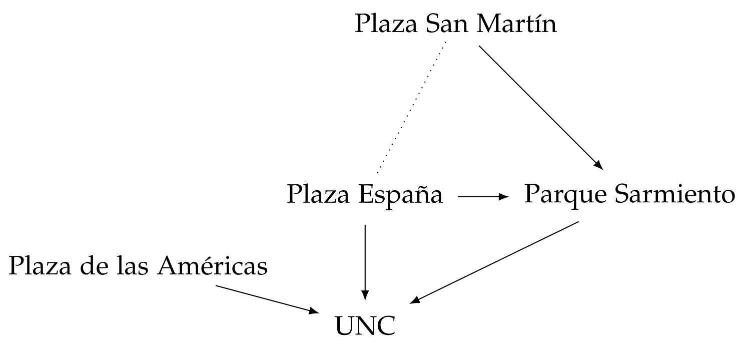


Figure 2: Graph representing a strike affecting the bus lines in Córdoba City center.

If we knew in advance about the partial strike, we could plan alternative routes to arrive to work. We know that some parts of the path from home to work have been sabotaged. If the usual bus stop is cut out from the bus line, we need to find an alternative from where we can continue our route to work. The situation in which we need to find an alternative way to go somewhere, can be represented introducing a "benevolent" factor, who adds edges that did not exist before. These new edges help us go to a previously isolated part of the graph (for instance, walking) where the buses work. In the following situation dotted lines are paths traversed by walk (formally, new edges added to the original model).

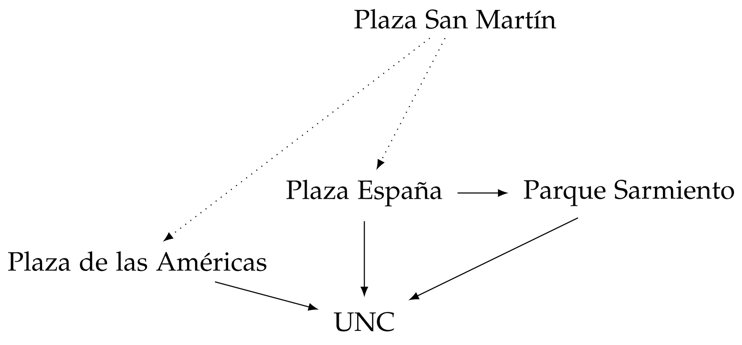


Figure 3: Graph representing alternative routes on walk.

It should be clear at this point the kind of dynamic scenarios we are interested in. What is important is that we can *formalize* all these situations and represent them in a dynamic framework. The most obvious way to do this would be by representing the whole space of possible evolutions as a graph, but this looks unwieldy. Instead, we can model and get mathematical tools to dynamically manage these situations. We are not only thinking in simple problems such as strikes in bus services or discovering a new path: we are thinking in abstract representations of this kind of situations, and a dynamic framework to deal with them. In this case, we can take some mathematical objects (graphs), and then design tools to study formal properties in this framework. Modal languages are suitable to describe properties of graphs, and we can also include operators that capture the possible evolutions of the graphs. As we explained in Section 2.2 it is possible to add new operators to handle dynamic behaviour, which shows us the real power of modal logics:

> *We can take a problem, think about the most appropriate way to model it and turn it into a modal logic.*

In Chapter 2 we discussed some examples of logics that can be used to model dynamic scenarios. In what follows, we will focus in situations in which we need to change the accessibility relation of a model. Modal logics let us explore the internal properties of the models, by inspecting the properties of the successors of a given evaluation point. These successors are reachable through the accessibility relation, which represents the *access* to the rest of the model. Other model modifiers have been investigated in previous work, but arbitrarily *changing the access* has not been studied yet. Logics with relation modifiers are used in different scenarios, and it is interesting to study this kind of languages from an abstract perspective to know more about their behaviour. For instance, in the field of epistemic logics, there are logics with operators to delete or add edges depending on a condition, which characterizes a set of states that we would like to connect or disconnect in the model. Some examples are Arrow Update Logic [Kooi and Renne, 2011a], Graph Modifiers [Aucher *et al.*, 2009] or Dynamic Epistemic Logics [van Ditmarsch *et al.*, 2007]. In our approach, we will first study arbitrary relation-changing operations, including new primitives and investigate their theoretical properties, such as expressive power, complexity of reasoning tasks, etc. In the next section we will introduce operators to delete, add and swap edges in a model, and we will start describing some properties.

3.2 INTRODUCING THE PRIMITIVES

Let us start by introducing the syntax.

Definition 3.2.1 (Syntax of Relation-Changing Modal Logics). *Let* PROP *be a countable, infinite set of propositional symbols. Then the set* FORM *of formulas over* PROP *is defined as:*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \blacklozenge\varphi,$$

where $p \in \mathsf{PROP}$, $\blacklozenge \in \{\Diamond, \langle \mathsf{sw} \rangle, \langle \mathsf{sb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{gbr} \rangle\}$ and $\varphi, \psi \in \mathsf{FORM}$. Other operators are defined as usual. In particular, $\blacksquare\varphi$ is defined as $\neg\blacklozenge\neg\varphi$.

We call $\mathcal{ML}(\blacklozenge)$ the extension of $\mathcal{ML}$ allowing also the $\blacklozenge$ operator, for $\blacklozenge \in \{\langle \mathsf{sw} \rangle, \langle \mathsf{sb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{gbr} \rangle\}$.

Formulas of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, $\mathcal{ML}(\langle \mathsf{sw} \rangle)$, $\mathcal{ML}(\langle \mathsf{br} \rangle)$, $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$, and $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ are evaluated in standard relational models, and the meaning of the operators of the basic modal logic is unchanged. When we evaluate formulas containing relation-changing operators, we will need to keep track of the edges that have been modified. To that end, let us define precisely the models that we will use. In the rest of this thesis we will use wv as a shorthand for $\{(w, v)\}$ or (w, v). Context will always disambiguate the intended use.

Definition 3.2.2 (Models and Model Variants). *A model $\mathcal{M}$ is a triple $\mathcal{M} = \langle W, R, V \rangle$, where W is a non-empty set whose elements are called points or states; $R \subseteq W \times W$ is the accessibility relation; and $V : \mathsf{PROP} \to \mathcal{P}(W)$ is a valuation.*

Given a model $\mathcal{M} = \langle W, R, V \rangle$, we define the following notations for model variants:

(sabotaging)	$\mathcal{M}^-_S = \langle W, R^-_S, V \rangle$, *with* $R^-_S = R \backslash S$, $S \subseteq R$.
(swapping)	$\mathcal{M}^*_S = \langle W, R^*_S, V \rangle$, *with* $R^*_S = (R \backslash S^{-1}) \cup S$, $S \subseteq R^{-1}$.
(bridging)	$\mathcal{M}^+_B = \langle W, R^+_B, V \rangle$, *with* $R^+_B = R \cup B$, $B \subseteq (W \times W) \backslash R$.

Let w be a state in $\mathcal{M}$, the pair $(\mathcal{M}, w)$ is called a pointed model; we will usually drop parentheses and call $\mathcal{M}, w$ a pointed model.

Model variants are Kripke models in which some updates have been done by dynamic operations. This notation will be a practical form to present the semantics of the new operators, and it will be also important later, when we introduce other constructions such as bisimulations or tableaux.

Definition 3.2.3 (Semantics)**.** *Given a pointed model $\mathcal{M}, w$ and a formula φ we say that $\mathcal{M}, w$ satisfies φ, and write $\mathcal{M}, w \models \varphi$, when*

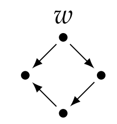
$$\begin{array}{lll}
\mathcal{M}, w \models \bot & & \textit{never} \\
\mathcal{M}, w \models p & \textit{iff} & w \in V(p) \\
\mathcal{M}, w \models \neg\varphi & \textit{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \textit{iff} & \mathcal{M}, w \models \varphi \textit{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \Diamond\varphi & \textit{iff} & \textit{for some } v \in W \textit{ s.t. } (w,v) \in R, \mathcal{M}, v \models \varphi \\
\mathcal{M}, w \models \langle \mathsf{sb} \rangle\varphi & \textit{iff} & \textit{for some } v \in W \textit{ s.t. } (w,v) \in R, \mathcal{M}^-_{wv}, v \models \varphi \\
\mathcal{M}, w \models \langle \mathsf{gsb} \rangle\varphi & \textit{iff} & \textit{for some } v, u \in W, \textit{ s.t. } (v,u) \in R, \mathcal{M}^-_{vu}, w \models \varphi \\
\mathcal{M}, w \models \langle \mathsf{sw} \rangle\varphi & \textit{iff} & \textit{for some } v \in W \textit{ s.t. } (w,v) \in R, \mathcal{M}^*_{vw}, v \models \varphi \\
\mathcal{M}, w \models \langle \mathsf{gsw} \rangle\varphi & \textit{iff} & \textit{for some } v, u \in W, \textit{ s.t. } (v,u) \in R, \mathcal{M}^*_{uv}, w \models \varphi \\
\mathcal{M}, w \models \langle \mathsf{br} \rangle\varphi & \textit{iff} & \textit{for some } v \in W \textit{ s.t. } (w,v) \notin R, \mathcal{M}^+_{wv}, v \models \varphi \\
\mathcal{M}, w \models \langle \mathsf{gbr} \rangle\varphi & \textit{iff} & \textit{for some } v, u \in W, \textit{ s.t. } (v,u) \notin R, \mathcal{M}^+_{vu}, w \models \varphi.
\end{array}$$

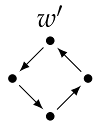
φ is satisfiable if for some pointed model $\mathcal{M}, w$ we have $\mathcal{M}, w \models \varphi$.

Relation-changing operators can delete, swap around or add new edges. The formula is then evaluated in the correspondent model variant. Notice that we extend the basic modal logic with primitives that let us perform these operations in two versions: local and global. The local version sabotages, swaps or bridges edges to adjacent nodes, while the global version modifies edges in any part of the model.

The semantic conditions for relation-changing operators look quite innocent but, as we will see in the next examples, these logics are actually very expressive. Besides the obvious effects of the modifiers, there are situations in which their semantics allows us to do something else, for instance, counting the number of accessible edges.

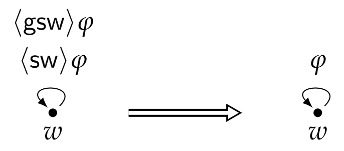
Example 3.2.4. *The $\langle \mathsf{sb} \rangle$ operator has no effect on acyclic models (its behaviour is exactly as a traditional $\Diamond$, as in the leftmost model). But in models containing cycles, $\langle \mathsf{sb} \rangle$ can count precisely the number of accessible edges by deleting each of them. In the rightmost model, after evaluating $\Diamond^5\top \wedge \langle \mathsf{sb} \rangle^4\Box\bot$ starting from w', we first go to some state at depth five, but with the second conjunct we return to w' destroying and counting four edges.*



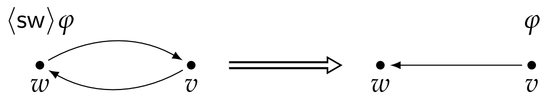


Example 3.2.5. *This example is extracted from [Rohde, 2006], and shows the expressivity of $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$. Consider the formula $[\mathsf{gsb}]\Diamond\top \wedge \langle \mathsf{gsb} \rangle\langle \mathsf{gsb} \rangle\Box\bot$. The first conjunct expresses that, if an arbitrary edge is deleted, then the evaluation point still has an outgoing edge regardless of the removed one. The second conjunct says that there are two edges that can be removed and such that the current position has no longer an outgoing edge. Thus, for any pointed model $\mathcal{M}, w$, we have $\mathcal{M}, w \models [\mathsf{gsb}]\Diamond\top \wedge \langle \mathsf{gsb} \rangle\langle \mathsf{gsb} \rangle\Box\bot$ if and only if the point w has exactly two distinct successors.*

Example 3.2.6. *The $\langle \mathsf{sw} \rangle$ and $\langle \mathsf{gsw} \rangle$ operators leave reflexive edges unchanged:*



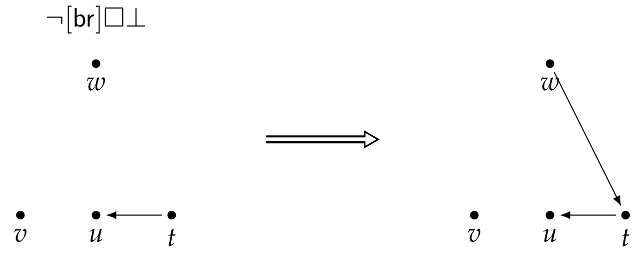
Example 3.2.7. *The* ⟨sw⟩ *operator can collapse symmetric edges into a single one:*



We start with the model on the left, where $R = \{wv, vw\}$ and evaluate $\langle\mathsf{sw}\rangle\varphi$ *at w. This implies evaluating φ at v after the relation is updated to $R^*_{vw} = (R \setminus wv) \cup vw = \{vw\}$, as shown on the right. This is actually the only situation where evaluating a* ⟨sw⟩ *formula leads to a model variant where the size of the accessibility relation R decreases.*

The same happens with the ⟨gsw⟩ *operator, when it swaps a symmetric edge.*

Example 3.2.8. *The* ⟨br⟩ *operator allows us to reach isolated parts of the model:*



Starting from w, which is a point with no successors, by evaluating $\neg[\mathsf{br}]\Box\bot$ *it is possible to reach an isolated part of the model and continue the evaluation of the rest of the formula.* $[\mathsf{br}]\Box\bot$ *establishes that all the points that have no links from w, have no successors. The formula is not satisfied, since there is no link from w to t and t has a successor (in the figure we add the link from w to t, and checking that u is reachable from t).*

3.3 SOME MODEL PROPERTIES

In this section, we will investigate some classical properties for modal logics. Some expressivity results are proved by showing that formulas of a certain language can or cannot enforce models with a determined structure. The Tree and Finite Model Property are examples of this kind of properties. The tree model property establishes that every satisfiable formula, is also satisfied at the root of a tree. The finite model property says that no formula can enforce an infinite model. $\mathcal{ML}$ has both properties. We will investigate what happens when we add relation-changing operators to the language.

Definition 3.3.1. *Given a language $\mathcal{L}$, we say that $\mathcal{L}$ has the* Tree Model Property *if for all $\mathcal{L}$-formula φ satisfiable, φ is satisfied in the root of a tree.*

Then we can state:

Theorem 3.3.2. *$\mathcal{ML}$ has the tree model property.*

Proof. In [Blackburn and van Benthem, 2006] a construction called *tree unraveling* is introduced. This construction was introduced first in [Dummet and Lemmon, 1959] and used in [Sahlqvist, 1973].

To unravel a model $\mathcal{M} = \langle W, R, V \rangle$ from a point w, take all finite sequences of points that start at w, which are reachable via R. Each sequence starting by w and ending by some v, has exactly the same valuation as v. These sequences form a tree with one-step extensions of sequences as the tree-successor relation. We call the model resultant from the tree unraveling of $\mathcal{M}$ as $Unr(\mathcal{M})$. Figure 4 shows an example of a tree unraveling.

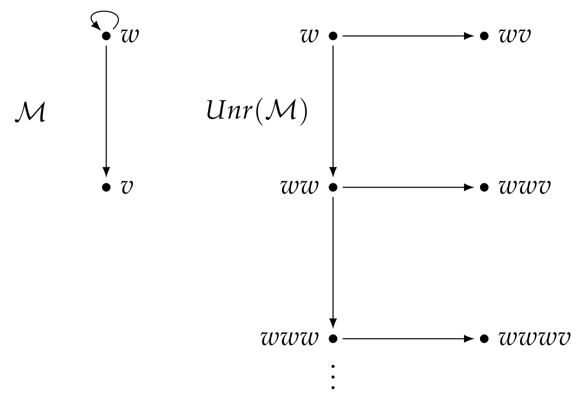


Figure 4: Example of a model and its tree unraveling

We will prove by structural induction, that for every pointed model $\mathcal{M}, w$, $\mathcal{ML}$-formula φ and sequence of points αw, $\mathcal{M}, w \models \varphi$ iff $Unr(\mathcal{M}), \alpha w \models \varphi$.

$\boldsymbol{\varphi = p}$: It is trivial, given that w and αw have the same valuation, for all α.

$\boldsymbol{\varphi = \neg\psi}$ and $\boldsymbol{\varphi = \psi \wedge \chi}$: These cases hold by inductive hypothesis.

$\boldsymbol{\varphi = \Diamond\psi}$: Suppose $\mathcal{M}, w \models \Diamond\psi$. Then, there exists v such that $R(w, v)$ and $\mathcal{M}, v \models \psi$. By inductive hypothesis, there is a sequence β, such that $Unr(\mathcal{M}), \beta v \models \psi$. Given the procedure to build the unraveling, some of these sequences β has to be αw, for some α, because w is a predecessor of v in the original model $\mathcal{M}$. Hence, $Unr(\mathcal{M}), \alpha w v \models \psi$, then $Unr(\mathcal{M}), \alpha w \models \Diamond\psi$.

□

We are interested in the behaviour of the logics introduced in Section 3.2. The question is: *"Adding relation-changing operators, do we increase the expressivity of the basic modal language?"*. We will introduce tools to study expressive power in the next chapter, but we can get already the first results. As we will see in the following theorem, by adding any of the relation-changing operators we introduced before, we lose the tree model property.

Theorem 3.3.3. *$\mathcal{ML}(\blacklozenge)$ does not have the tree model property, for $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$.*

Proof. We present formulas that ensure that the accessibility relation does not define a tree. For $\langle \mathsf{gsb} \rangle$, the result has already been proved in [Löding and Rohde, 2003a], for $\langle \mathsf{sw} \rangle$ in [Areces *et al.*, 2013b] and for $\langle \mathsf{sb} \rangle$ and $\langle \mathsf{br} \rangle$ in [Areces *et al.*, 2012]. Let us define some notation that will be helpful in this proof, and in the rest of the thesis:

- $\blacksquare^0 \varphi = \varphi$,
- $\blacksquare^{n+1} \varphi = \blacksquare \blacksquare^n \varphi$,
- $\blacksquare^{(n)} \varphi = \bigwedge_{1 \le i \le n} \blacksquare^i \varphi$,

for $\blacksquare \in \{\Box, [\mathsf{sb}], [\mathsf{gsb}], [\mathsf{sw}], [\mathsf{gsw}], [\mathsf{br}], [\mathsf{gbr}]\}$.

Suppose the following formulas hold at some point w in a model:

1. $\Diamond\Diamond\top \ \wedge \ [\mathsf{sb}]\Box\bot$, then w is reflexive;
2. $\Diamond\Diamond\top \ \wedge \ [\mathsf{gsb}]\Box\bot$, same;
3. $p \wedge \Box^{(3)}\neg p \wedge \langle \mathsf{sw} \rangle \Diamond\Diamond p$, then w has a reflexive successor;
4. $\Box\bot \wedge \langle \mathsf{gsw} \rangle \Diamond\top$, then w has an incoming edge.
5. $\Box\bot \wedge \langle \mathsf{br} \rangle \langle \mathsf{br} \rangle \top$, then w and some different point v are unconnected;
6. $\Box\bot \wedge \langle \mathsf{gbr} \rangle \langle \mathsf{gbr} \rangle \top$, same.

In each case, the formula cannot be satisfied in the root of a tree. We will now discuss each case in more detail.

1. The formula
$$\varphi = \Diamond\Diamond\top \ \wedge \ [\mathsf{sb}]\Box\bot$$
is true at a state w in a model, only if w is reflexive.

   Suppose we evaluate φ at some state w of an arbitrary model. On the one hand, the "static" part of the formula $\Diamond\Diamond\top$ ensures it is possible to take two steps on the accessibility relation. On the other hand, the "dynamic" part of the formula $[\mathsf{sb}]\Box\bot$ tells us that after taking any accessible edge and eliminating it, it is no longer possible to go anywhere else. This can only happen if the point w is reflexive and does not have any other outgoing edges.

2. Similarly, the formula
$$\varphi = \Diamond\Diamond\top \ \wedge \ [\mathsf{gsb}]\Box\bot$$
(from [Löding and Rohde, 2003a]) is true at a state w in a model, only if w is reflexive.

3. The formula
$$\varphi = p \wedge \Box^{(3)}\neg p \wedge \langle \mathsf{sw} \rangle \Diamond\Diamond p$$
is true at a state w in a model, only if w has a reflexive successor.

   Suppose we evaluate φ at some state w of an arbitrary model. The "static" part of the formula $(p \wedge \Box^{(3)}\neg p)$ makes sure that p is true in w and that no p-state is reachable within three steps from w (in particular, w cannot be reflexive).

   Because $\langle \mathsf{sw} \rangle \Diamond\Diamond p$ is true at w, there should be an R-successor v where $\Diamond\Diamond p$ holds once the accessibility relation has been updated to R^*_{vw}. That is, v has to reach a p-state in exactly two R^*_{vw}-steps. But the only p-state sufficiently close is w which is reachable in one step. As w is not reflexive, v has to be reflexive so that we can linger at v for one loop and reach p in the correct number of states.

4. The formula
$$\Box\bot \wedge \langle \mathsf{gsw} \rangle \Diamond\top$$
is true at a state w in a model, only if w has an incoming edge.

5. The formula
$$\varphi = \Box\bot \wedge \langle \mathsf{br} \rangle \langle \mathsf{br} \rangle \top$$
is only satisfiable in models that have at least two points. These points are unconnected or the evaluation point has an incoming edge.

6. Similarly, the formula
$$\varphi = \Box\bot \wedge \langle \mathsf{gbr} \rangle \langle \mathsf{gbr} \rangle \top$$
is only satisfiable in models that have at least two unconnected points.

□

The previous results give us the first intuitions on the expressivity of relation-changing modal logics. We showed in Theorem 3.3.3 that there are satisfiable formulas of the six logics that we investigate, which cannot be satisfied at the root of a tree. This shows that adding operators to change the accessibility relation we obtain more expressivity. The challenge is to discover how much expressive power we added. We will explore now, another classical property for modal logics: the finite model property. This property establishes that every satisfiable formula is satisfied in a finite model. $\mathcal{ML}$ has the finite model property, and it also satisfies a stronger variant: the bounded model property. It is called "bounded" because besides the finite model property, we can determine a bound for the finite models. Let introduce the formal definition.

Definition 3.3.4. *Given a language $\mathcal{L}$, we say that $\mathcal{L}$ has the* Finite Model Property *if for all $\mathcal{L}$-formula φ satisfiable, φ is satisfied in a finite model.*

$\mathcal{L}$ has the Bounded Finite Model Property *if for all $\mathcal{L}$-formula φ satisfiable, φ is satisfied in a finite model, whose size is bounded by a function on the size of the formula.*

Then, we can introduce next theorem for the basic modal logic:

Theorem 3.3.5. *$\mathcal{ML}$ has the bounded finite model property.*

Proof (Sketch). A complete proof via selection or filtration can be found in [Blackburn *et al.*, 2001]. We explain in a few words the argument of the proof via filtrations. Consider the satisfiability problem for Propositional Logic. We can check if a formula is satisfiable by inspecting truth tables, which are all the possible assignments of truth values to the propositional symbols that appear in the formula. If we consider the whole propositional language, we would need to assign values to an infinite set of variables, but it is not necessary to give a value to those variables that do not appear in the formula. Thereby, throwing away the "useless" symbols we need to evaluate a finite number of propositional variables.

In this case we collapse all the valuations that are equivalent by inspecting just the symbols in the formula. The filtration method for modal logics follows the same idea: given a formula φ and a model $\mathcal{M}$, a finite model is defined by collapsing to a single point all the points in $\mathcal{M}$ that satisfy the same subformulas of φ. The resulting model satisfies φ if and only if the original one does. The finite model obtained by filtrations has, at most $2^{s(\varphi)}$ states, where $s(\varphi)$ is the number of the subformulas of φ. As a result, we have a bound for the size of the finite model. $\square$

We will prove that the logics we introduced do not have the finite model property. As we will see in the following theorem, each of the six logics we are investigating has formulas that only hold in an infinite model.

Theorem 3.3.6. *$\mathcal{ML}(\blacklozenge)$ does not have the finite model property, for $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$.*

For each logic we give a formula that forces infinite models. Let first explain one case in detail and then introduce the rest of the formulas. We start with the case of $\mathcal{ML}(\langle \mathsf{sw} \rangle)$.

We will exhibit a formula φ, which is a conjunction of several properties that forces models to have an infinite chain of states. It does so by ensuring that the propositional symbol s is true only in the evaluation state w (named the spy point), and false in all the states reachable from w. The spy point sees all the points of the model, but the rest of the points cannot see the spy. Then, φ enforces specific properties on the model, locating states by their distance to w using formulas of the form $\Diamond \ldots \Diamond s$, after swapping an outgoing edge from w. In this way, it is possible to enforce seriality, irreflexivity and transitivity on a chain of states. The conjunction of these three properties can only be satisfied in an infinite model. The intended model is showed in Figure 5.

Let us see in detail, how each part of φ works to get the intended model. Suppose that φ is evaluated at some state w, where φ is the conjunction of the following formulas:

1. $s \wedge \Box^{(9)} \neg s$.
   This formula makes s true at state w and false at all states accessible within 9 steps.

2. $\Diamond \top$.
   It indicates that w has a successor.

3. $\Box \Diamond \top$.
   The formula says that every successor of w has some successor.

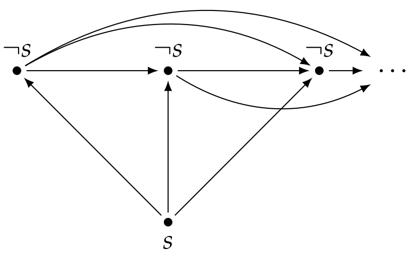


Figure 5: Infinite model for $\mathcal{ML}(\langle \mathsf{sw} \rangle)$.

4. $[\mathsf{sw}]\Box\neg\Diamond s$.
   This makes irreflexive every successor of the w. After swapping any outgoing edge from w and reaching some state v, all the successors of v only see states that do not satisfy s. Hence v has to be irreflexive, because as it was reached from w by swapping an edge, now it can see an s-state, precisely w.

5. $[\mathsf{sw}][\mathsf{sw}](\neg s \rightarrow \Diamond\Diamond\Diamond\Diamond\Diamond s)$.
   This formula tells that from any state $v \neq w$ reachable in two swapping steps, it is possible to go back to w in five steps. But this is only possible by first going to w in two steps, then going to v in one step and going again to w in two steps. Hence all states accessible in two steps from w are also accessible in one. This makes w a "spy point", i.e., it is directly connected to every state in the submodel generated from it.

6. $[\mathsf{sw}][\mathsf{sw}][\mathsf{sw}](\neg\Diamond s \rightarrow \Diamond\Diamond\Diamond(\neg s \wedge \Diamond\Diamond\Diamond s))$.
   Finally, it enforces the previous property on the successors of w.

(3), (4) and (6) respectively enforce seriality, irreflexivity and transitivity on a chain of states starting from w. Hence this chain must be infinite.

Now, we are ready to explore the complete proof of the lack of finite model property for the six relation-changing modal logics investigated in this thesis.

Proof. (of Theorem 3.3.6). Let us see the formulas that force infinite models in each case.

1. For $\mathcal{ML}(\langle \mathsf{sw} \rangle)$, we have the conjunction of the formulas described before.

$$\begin{array}{rlr}
\varphi = & s \wedge \Box^{(9)}\neg s & (1)\\
& \wedge\ \Diamond\top & (2)\\
& \wedge\ \Box\Diamond\top & (3)\\
& \wedge\ [\mathsf{sw}]\Box\neg\Diamond s & (Irr)\\
& \wedge\ [\mathsf{sw}][\mathsf{sw}](\neg s \rightarrow \Diamond\Diamond\Diamond\Diamond\Diamond s) & (Spy)\\
& \wedge\ [\mathsf{sw}][\mathsf{sw}][\mathsf{sw}](\neg\Diamond s \rightarrow \Diamond\Diamond\Diamond(\neg s \wedge \Diamond\Diamond\Diamond s)) & (Trans)
\end{array}$$

   As we said in the sketch preceding the proof, φ is satisfiable and, every model of the formula has to be infinite. The model introduced in Figure 5 is one satisfying φ.

2. For $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$.

$$\begin{array}{rll} \varphi = & s \wedge \Box^{(9)}\neg s \wedge \Diamond\top \wedge \Box\Diamond\top & (1) \\ & \wedge\ \Box\Box[\mathsf{gsw}][\mathsf{gsw}]\Box\Box(s \to \Diamond\Diamond\Diamond s) & (Spy) \\ & \wedge\ \Box[\mathsf{gsw}](\Diamond s \to \Box\Box\neg s) & (Irr) \\ & \wedge\ \Box\Box\Box[\mathsf{gsw}][\mathsf{gsw}][\mathsf{gsw}](\Diamond\Diamond\Diamond s \to \Diamond\Diamond\Diamond(\neg s \wedge \Diamond\Diamond\Diamond s)) & (Trans) \end{array}$$

The formula above is also satisfiable and their models are infinite, in a similar way as for $\mathcal{ML}(\langle \mathsf{sw} \rangle)$. The model in Figure 5 satisfies φ.

3. For $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ we follow the same ideas, but now the enforced models have also edges from any state to the spy point. In this case, we sabotage these edges to identify visited states. Figure 6 illustrates the model we want to enforce.

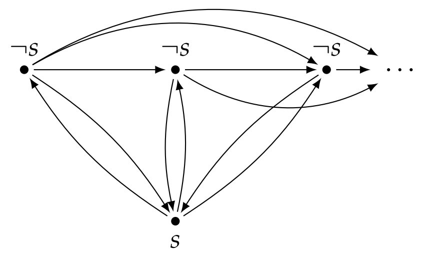


Figure 6: Infinite model for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

Now we introduce the formula that enforces infinite models as the one in Figure 6. After that, we describe more in detail how each part of the formula works.

$$\begin{array}{rll} \varphi = & s \wedge \Box\neg s \wedge \Diamond\top \wedge \Box\Diamond s & (1) \\ & \wedge\ \Box\Box(s \to \neg\Diamond s) & (2) \\ & \wedge\ [\mathsf{sb}][\mathsf{sb}](s \to \Box\Diamond s) & (3) \\ & \wedge\ \Box[\mathsf{sb}](s \to \Diamond\neg\Diamond s) & (4) \\ & \wedge\ \Box\Diamond\neg s & (5) \\ & \wedge\ \Box\Box(\neg s \to \Diamond(s \wedge \neg\Diamond s)) & (6) \\ & \wedge\ \Box[\mathsf{sb}](\neg s \to [\mathsf{sb}](s \to \Box\Box(\neg s \to \Diamond s))) & (7) \\ & \wedge\ \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\Diamond(\neg s \wedge \neg\Diamond s))) & (8) \\ & \wedge\ \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\neg\Diamond s)) & (Spy) \\ & \wedge\ \Box[\mathsf{sb}](s \to \Box(\neg\Diamond s \to \Box\Diamond s)) & (Irr) \\ & \wedge\ \neg\Diamond\langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \Diamond s \wedge \Diamond\neg\Diamond s))) & (3cyc) \\ & \wedge\ \Box\langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\neg\Diamond s))))) & (Trans) \end{array}$$

The first part of (1) establishes that s holds at the evaluation point and does not hold at any successor accessible from there in one step (in particular, the evaluation point cannot be reflexive). The second part of the formula ensures there is one successor, and each successor sees an s-point.

(2), (3) and (4) ensure that the states accessible in one step from the evaluation point, have a link back to it. In particular (2) says that each s-point reachable from the evaluation point in two steps cannot have a successor where s holds (in particular, this s-point cannot be reflexive).

(5) creates an infinite chain of elements, by ensuring that each successor has an edge to a point where $\neg s$ is true. The formulas (6), (7) and (8) play the same role as (2), (3) and (4), but ensuring that states accessible in two steps have a link back to the evaluation point.

(Spy) creates an spy-point: any point accessible in two steps from the evaluation point can also be accessible in one. (Irr) and $(Trans)$ enforce irreflexivity and transitivity respectively, and $(3cyc)$ ensures that there are not 3 elements of the infinite chain of states that form a cycle.

4. For $\mathcal{ML}(\langle \mathsf{gsb}\rangle)$ we build a formula that enforces infinite models like the one of Figure 7. It is similar to the one for $\mathcal{ML}(\langle \mathsf{sb}\rangle)$, except that (2) establishes that in one step, there is an s-successor which is unique, and (4) is the same as (2) but in two steps. The s-successor in two steps may be different from the spy point.

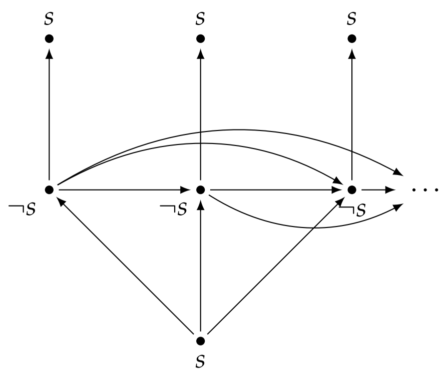


Figure 7: Infinite model for $\mathcal{ML}(\langle \mathsf{gsb}\rangle)$.

We introduce now the formula that can only be satisfied in an infinite model as the previous one.

$$\begin{array}{rll}
\varphi = & s \wedge \Box\neg s \wedge \Diamond\top & (1)\\
& \wedge\ \Box(\Diamond s \wedge \langle \mathsf{gsb}\rangle\neg\Diamond s) & (2)\\
& \wedge\ \Box\Diamond\neg s & (3)\\
& \wedge\ \Box\Box(\neg s \rightarrow \Diamond s \wedge \langle \mathsf{gsb}\rangle\neg\Diamond s) & (4)\\
& \wedge\ [\mathsf{gsb}](\ \Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \neg\Diamond s)) \ \rightarrow\ \Diamond\neg\Diamond s\ ) & (Spy)\\
& \wedge\ [\mathsf{gsb}]\Box(\neg\Diamond s \rightarrow \Box\Diamond s) & (Irr)\\
& \wedge\ [\mathsf{gsb}]\neg\Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \Diamond s \wedge \Diamond\neg\Diamond s)) & (3cyc)\\
& \wedge\ [\mathsf{gsb}][\mathsf{gsb}](\ \Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \neg\Diamond s)) \ \rightarrow\ \Diamond(\neg\Diamond s \wedge \Diamond\neg\Diamond s)\ ) & (Trans)
\end{array}$$

5. For $\mathcal{ML}(\langle \mathsf{br}\rangle)$ we generate infinite models with other structure. The idea now, is to create a "bag" of spy points, that are all related among them. Then, we can control the new edges that are created to some $\neg s$-state. One of those infinite models is shown in Figure 8.

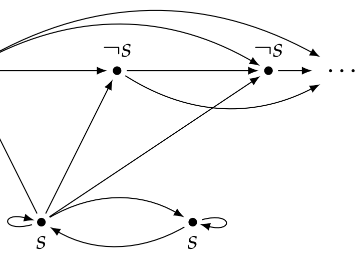


Figure 8: Infinite model for $\mathcal{ML}(\langle \mathsf{br} \rangle)$.

The formula which generates such a model is:

$$\begin{array}{rll} \varphi = & s \wedge [\mathsf{br}]\neg s & (1) \\ & \wedge\, \Box[\mathsf{br}]\neg s & (s - connex) \\ & \wedge\, \Box\Box s & (3) \\ & \wedge\, \langle \mathsf{br} \rangle \top & (4) \\ & \wedge\, \Box[\mathsf{br}]\Box\neg s & (5) \\ & \wedge\, [\mathsf{br}]\Diamond\top & (Ser) \\ & \wedge\, [\mathsf{br}][\mathsf{br}](s \to \Box(\neg s \to \Box(\neg s \to \Box\neg s))) & (Irr) \\ & \wedge\, [\mathsf{br}]\Box\Box[\mathsf{br}](s \to \Diamond(\neg s \wedge \Diamond\Diamond s)) & (Trans) \end{array}$$

6. For $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$ we enforce the same kind of models as for $\mathcal{ML}(\langle \mathsf{br} \rangle)$, for instance, model of Figure 8.

$$\begin{array}{rll} \varphi = & s \wedge \Box\neg s \wedge \Diamond\top \wedge \Box\Box\neg s & (1) \\ & \Box\Diamond\neg s & (2) \\ & \Box\Box(\neg s \to \Box\neg s) & (3) \\ & [\mathsf{gbr}](\ \Diamond(\neg\Diamond s \wedge \Diamond\Diamond s)\ \to\ \Diamond\Diamond s\ ) & (Spy) \\ & [\mathsf{gbr}]\Box(\Diamond s \to \Box\neg\Diamond s) & (Irr) \\ & [\mathsf{gbr}]\neg\Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \Diamond(\neg s \wedge \neg\Diamond s \wedge \Diamond\Diamond s))) & (3cyc) \\ & [\mathsf{gbr}][\mathsf{gbr}](\ \Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \Diamond(\neg s \wedge \Diamond s)))\ \to\ \Diamond(\Diamond s \wedge \Diamond\Diamond s)\ ) & (Trans) \end{array}$$

□

We showed the lack of the tree and the finite model property for relation-changing modal logics. We found very simple formulas that cannot be satisfied at the root of a tree-like model. Enforcing infinite models was more difficult. The main idea to do this is the use of a *spy point technique*. A spy point is a state of the model which can access any other state in the model. In this section, we used different kind of spy points according to the expressivity of each language. For instance, in the cases of bridge operators, we were not able to guarantee the uniqueness of a spy point, instead we use a set of spy points. After that, we follow the same ideas: formulas enforce serial, irreflexive and transitive models, which implies that the models are infinite.

We have seen that the six logics we are investigating are more expressive than basic modal logic. However, we have not established their exact boundaries. In the next chapter we will introduce the appropriate tools to investigate the expressivity of these languages: *bisimulations* and *Ehrenfeucht-Fraïssé games*. We will start by introducing the notions for $\mathcal{ML}$, and we will extend the results to investigate relation-changing modal logics.

4 Expressive Power

Anything that thinks logically can be fooled by something else that thinks at least as logically as it does.

from "The Hitchhiker's Trilogy", Douglas Adams.

4.1 MODAL DISTINCTIONS

In previous chapters, we have used different tools to discuss expressive power of different languages. For instance, we used *translations* to prove that $\mathcal{ML}$ is a fragment of $\mathcal{FOL}^2$, and we did the same for other modal operators. For relation-changing operators, we proved the lack of some *model properties* by enforcing certain kind of structures, to show that the six logics we introduced are more expressive than $\mathcal{ML}$. All these results say something about the expressive power of the languages, but if we want to study in detail these languages we need more specific tools. The expressive power of a language is measured in terms of the distinctions we can draw with it. For this, *bisimulations* introduced by van Benthem in [van Benthem, 1984; van Benthem, 1985] will be helpful.

In modern modal model theory, the notion of bisimulation is a crucial tool. Typically, a bisimulation is a binary relation linking elements of the domains that have the same atomic information, and preserving the relational structure of the model. Let us introduce the notion of bisimulation for the basic modal language $\mathcal{ML}$.

Definition 4.1.1 ($\mathcal{ML}$-Bisimulations). *Let $\mathcal{M} = \langle W, R, V\rangle$, $\mathcal{M}' = \langle W', R', V'\rangle$ be two models. A non empty relation $Z \subseteq W \times W'$ is an $\mathcal{ML}$-bisimulation if it satisfies the following conditions. If wZw' then*

(atomic harmony) *for all $p \in \mathsf{PROP}$, $w \in V(p)$ iff $w' \in V'(p)$;*

(zig) *if $(w, v) \in R$ then for some v', $(w', v') \in R'$ and vZv';*

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

Given two pointed models $\mathcal{M}, w$ and $\mathcal{M}', w'$ we say that they are $\mathcal{ML}$-bisimilar and we write $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}} \mathcal{M}', w'$ if there is an $\mathcal{ML}$-bisimulation Z such that wZw'.

The importance of the notion of bisimulation can be seen in the following theorem, that establishes that bisimulations relate models that are modally equivalent.

Theorem 4.1.2 (Invariance Under Bisimulations). *Let $\mathcal{M}=\langle W, R, V\rangle$, $\mathcal{M}'=\langle W', R', V'\rangle$ be two models, $w \in W$ and $w' \in W'$. If there is an $\mathcal{ML}$-bisimulation Z between $\mathcal{M}, w$ and $\mathcal{M}', w'$ such that wZw' then for any formula $\varphi \in \mathcal{ML}$, $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}', w' \models \varphi$.*

Proof. The proof is by structural induction on $\mathcal{ML}$-formulas.

$\varphi = p$: Suppose $\mathcal{M}, w \models p$. Then by definition of $\models$, $w \in V(p)$. But as wZw' and by (atomic harmony), $w' \in V'(p)$, then $\mathcal{M}', w' \models p$.

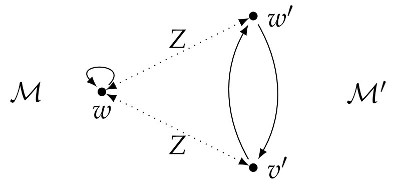
$\varphi = \neg\psi$: Suppose $\mathcal{M}, w \models \neg\psi$. Then by definition of $\models$, $\mathcal{M}, w \not\models \psi$. But then, by I.H., $\mathcal{M}', w' \not\models \psi$ then $\mathcal{M}', w' \models \neg\psi$.

$\varphi = \psi \wedge \chi$: Suppose $\mathcal{M}, w \models \psi \wedge \chi$. Then $\mathcal{M}, w \models \psi$ and $\mathcal{M}, w \models \chi$. By I.H., $\mathcal{M}', w' \models \psi$ and $\mathcal{M}', w' \models \chi$, then $\mathcal{M}', w' \models \psi \wedge \chi$.

$\varphi = \Diamond\psi$: Suppose $\mathcal{M}, w \models \Diamond\psi$. Then there is v in W s.t. $R(w, v)$ and $\mathcal{M}, v \models \psi$. By (zig) we have v' in W' such that $R'(w', v')$ and vZv'. By I.H., $\mathcal{M}', v' \models \psi$ and by definition $\mathcal{M}, w' \models \Diamond\psi$. For the other direction use (zag).

□

Example 4.1.3. *This is an example of two models that cannot be distinguished by any formula of $\mathcal{ML}$. Dotted lines represent the relation Z, which defines a bisimulation.*



We have presented bisimulations in a relational perspective, and we showed that $\mathcal{ML}$ cannot distinguish between bisimilar models. But they can also be presented from a dynamic perspective. Checking whether two models are bisimilar can be recast in terms of *Ehrenfeucht-Fraïssé Games* [Ebbinghaus *et al.*, 1984; Sangiorgi, 2009].

Definition 4.1.4 (Ehrenfeucht-Fraïssé Games). *Let $\mathcal{M}, w$ and $\mathcal{M}', w'$ be two pointed models.*

An Ehrenfeucht-Fraïssé game *$EF(\mathcal{M}, \mathcal{M}', w, w')$ is defined as follows. There are two players called* Spoiler *and* Duplicator*. Duplicator immediately loses if w and w' do not agree (they do not satisfy the same 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.*

Spoiler chooses v, a successor of w. If w has no successors, then Duplicator wins. Duplicator has to choose v', a successor of w', such that v and v' agree. If there is no such successor, Spoiler wins. Otherwise the game continues with $EF(\mathcal{M}, \mathcal{M}', v, v')$. If Spoiler starts by choosing a successor v' of w', the same process has to be followed by exchanging the models where each player has to choose.

The winning conditions for the game $EF(\mathcal{M}, \mathcal{M}', w, w')$ establishes that before the game begins, Duplicator immediately loses if w and w' do not coincide in the propositional symbols. In subsequent rounds, if Duplicator responds with a successor that differs in the atomic propositions with respect to the point chosen by Spoiler, Duplicator loses. If one player cannot move, the other wins, and Duplicator wins on infinite runs of which Spoiler does not win. We also assume that both players always make a move if they can. Given these conditions, observe that exactly one of Spoiler or Duplicator wins each game.

Given two pointed models $\mathcal{M}, w$ and $\mathcal{M}', w'$ we will write $\mathcal{M}, w \equiv^{EF} \mathcal{M}', w'$ when Duplicator has a winning strategy for $EF(\mathcal{M}, \mathcal{M}', w, w')$.

Theorem 4.1.5. *Given two pointed models $\mathcal{M}, w$ and $\mathcal{M}', w'$ then $\mathcal{M}, w \equiv^{EF} \mathcal{M}', w'$ if and only if $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}} \mathcal{M}', w'$.*

Proof. For a full proof of this theorem, see [Goranko and Otto, 2005]. □

For instance, let see Example 4.1.3 starting from w and w'. If Spoiler starts in $\mathcal{M}$, the only choice to pick a successor is w itself. Then Duplicator can move to v' in $\mathcal{M}'$ as an answer. If Spoiler picks again w, Duplicator can return to w'. On the other hand, for any choice that Spoiler can do in $\mathcal{M}'$, Duplicator always has w in $\mathcal{M}$ to answer. In this way we can see that Duplicator has a winning strategy for this game, then $\mathcal{M}, w \equiv^{EF} \mathcal{M}', w'$, which is we expect given that $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}} \mathcal{M}', w'$.

Theorem 4.1.2 says that bisimilar models are modally equivalent. This is a very important result because we can replace a model for a bisimilar one and the logic will not note it. On the other hand, Theorem 4.1.5 establishes that bisimilarity and *EF*-game equivalence are the same. From these two results we can infer than *EF*-game equivalence also implies modal equivalence. Now we have an operative way to check when two models are indistinguishable. However, for relation-changing modal logics this notion is not enough. Bisimulations as we presented in this section do not capture the behaviour of the dynamic operators. Fortunately, we do not need to find new tools to study relation-changing modal logics. In the next section, we will show that adapting the existent tools, we capture exactly the meaning of all the six relation-changing modal logics investigated in this thesis.

4.2 THE RIGHT WAY TO DESCRIBE THINGS

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?"*. As we already mentioned, in modal logics the tool to formalize this notion is *bisimulation*. Previously we defined bisimulations as relations between states of the models. In logics we introduced in Chapter 3, just relating elements of the domains is not enough. Because we are studying logics where a model can change in any moment while we are evaluating a formula, we need to keep track of the modifications that the model already suffered. For relation-changing modal logics, bisimulations capture the dynamic behaviour linking states together with the current accessibility relation. In [Areces *et al.*, 2012; Areces *et al.*, 2013b] we introduced the following notions of bisimulations:

Definition 4.2.1 ($\mathcal{ML}(\blacklozenge)$-Bisimulations)**.** *Let $\mathcal{M} = \langle W, R, V\rangle$, $\mathcal{M}' = \langle W', R', V'\rangle$ be two models. A non empty relation $Z \subseteq (W \times \mathcal{P}(W^2)) \times (W' \times \mathcal{P}(W'^2))$ is an $\mathcal{ML}(\blacklozenge)$-bisimulation if it satisfies the conditions* atomic harmony, zig *and* zag *below, and the corresponding conditions for the operators that the considered logic contains. If $(w, S)Z(w', S')$ then*

(atomic harmony) *for all $p \in$ PROP, $w \in V(p)$ iff $w' \in V'(p)$;*

(zig) *if* $(w, v) \in S$ *then for some* v', $(w', v') \in S'$ *and* $(v, S)Z(v', S')$*;*

(zag) *if* $(w', v') \in S'$ *then for some* v, $(w, v) \in S$ *and* $(v, S)Z(v', S')$*;*

($\langle\mathsf{sb}\rangle$-zig) *if* $(w, v) \in S$ *then for some* v', $(w', v') \in S'$ *and* $(v, S^{-}_{vw})Z(v'S'^{-}_{v'w'})$*;*

($\langle\mathsf{sb}\rangle$-zag) *if* $(w', v') \in S'$ *then for some* v, $(w, v) \in S$ *and* $(v, S^{-}_{wv})Z(v'S'^{-}_{w'v'})$*;*

($\langle\mathsf{gsb}\rangle$-zig) *if* $(u, v) \in S$ *then for some* u', v', $(u', v') \in S'$ *and* $(w, S^{-}_{uv})Z(w'S'^{-}_{u'v'})$*;*

($\langle\mathsf{gsb}\rangle$-zag) *if* $(u', v') \in S'$ *then for some* u, v, $(u, v) \in S$ *and* $(w, S^{-}_{uv})Z(w'S'^{-}_{u'v'})$*;*

($\langle\mathsf{sw}\rangle$-zig) *if* $(w, v) \in S$ *then for some* v', $(w', v') \in S'$ *and* $(v, S^{*}_{vw})Z(v'S'^{*}_{v'w'})$*;*

($\langle\mathsf{sw}\rangle$-zag) *if* $(w', v') \in S'$ *then for some* v, $(w, v) \in S$ *and* $(v, S^{*}_{vw})Z(v'S'^{*}_{v'w'})$*;*

($\langle\mathsf{gsw}\rangle$-zig) *if* $(u, v) \in S$ *then for some* u', v', $(u', v') \in S'$ *and* $(w, S^{*}_{vu})Z(w'S'^{*}_{v'u'})$*;*

($\langle\mathsf{gsw}\rangle$-zag) *if* $(u', v') \in S'$ *then for some* u, v, $(w, v) \in S$ *and* $(w, S^{*}_{vu})Z(w'S'^{*}_{v'u'})$*;*

($\langle\mathsf{br}\rangle$-zig) *if* $(w, v) \notin S$*, there is* $v' \in W'$ *s.t.* $(w', v') \notin S'$ *and* $(v, S^{+}_{wv})Z(v', S'^{+}_{w'v'})$*;*

($\langle\mathsf{br}\rangle$-zag) *if* $(w', v') \notin S'$*, there is* $v \in W$ *s.t.* $(w, v) \notin S$ *and* $(v, S^{+}_{wv})Z(v', S'^{+}_{w'v'})$*;*

($\langle\mathsf{gbr}\rangle$-zig) *if* $(u, v) \notin S$*, there is* $u', v' \in W'$ *s.t.* $(u', v') \notin S'$ *and* $(w, S^{+}_{uv})Z(w', S'^{+}_{u'v'})$*;*

($\langle\mathsf{gbr}\rangle$-zag) *if* $(u', v') \notin S'$*, there is* $u, v \in W$ *s.t.* $(u, v) \notin S$ *and* $(w, S^{+}_{uv})Z(w', S'^{+}_{u'v'})$*.*

Given two pointed models $\mathcal{M}, w$ *and* $\mathcal{M}', w'$ *we say that they are* $\mathcal{ML}(\blacklozenge)$*-bisimilar and we write* $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}(\blacklozenge)} \mathcal{M}', w'$ *if there is an* $\mathcal{ML}(\blacklozenge)$*-bisimulation* Z *such that* $(w, R)Z(w', R')$ *where* R *and* R' *are respectively the relations of* $\mathcal{M}$ *and* $\mathcal{M}'$*.*

As we have proved for $\mathcal{ML}$, bisimulations are important to distinguish when two models are equal for those languages. The next theorem establishes that two bisimilar models are not distinguishable for any formula of the corresponding language.

Theorem 4.2.2 (Invariance Under Bisimulations)**.** *Let* $\mathcal{M} = \langle W, R, V\rangle$ *and* $\mathcal{M}' = \langle W', R', V'\rangle$ *be two models,* $w \in W$, $w' \in W'$*, and let* $S \subseteq W^2, S' \subseteq W'^2$*. If there is an* $\mathcal{ML}(\blacklozenge)$*-bisimulation* Z *between* $\mathcal{M}, w$ *and* $\mathcal{M}', w'$ *such that* $(w, S)Z(w', S')$ *then for any formula* $\varphi \in \mathcal{ML}(\blacklozenge)$*,* $\langle W, S, V\rangle, w \models \varphi$ *iff* $\langle W', S', V'\rangle, w' \models \varphi$*.*

Proof. We will see the case for $\mathcal{ML}(\langle\mathsf{sw}\rangle)$. The proof is by structural induction on $\mathcal{ML}(\langle\mathsf{sw}\rangle)$-formulas. The base case holds by (atomic harmony), and the $\wedge$ and $\neg$ cases are trivial.

$\boldsymbol{\varphi = \Diamond\psi}$: Suppose $\langle W, S, V\rangle, w \models \Diamond\psi$. Then there is v in W s.t. $(w, v) \in S$ and $\langle W, S, V\rangle, v \models \psi$. By (zig) we have v' in W' such that $w'S'v'$ and $(v, S)Z(v', S')$. By I.H., $\langle W', S', V'\rangle', v' \models \psi$ and by definition $\langle W', S', V'\rangle, w' \models \Diamond\psi$. For the other direction use (zag).

$\boldsymbol{\varphi = \langle\mathsf{sw}\rangle\psi}$: For the left to the right direction suppose $\langle W, S, V\rangle, w \models \langle\mathsf{sw}\rangle\psi$. Then there is v in W s.t. $(w, v) \in S$ and $\langle W, S^{*}_{vw}, V\rangle, v \models \psi$. By ($\langle\mathsf{sw}\rangle$-zig) we have v' in W' s.t. $(w', v') \in S'$ and $(v, S^{*}_{vw})Z(v', S'^{*}_{v'w'})$. By I.H., $\langle W', S'^{*}_{v'w'}, V'\rangle, v' \models \psi$ and by definition $\langle W', S', V'\rangle, w' \models \langle\mathsf{sw}\rangle\psi$. For the other direction use ($\langle\mathsf{sw}\rangle$-zag). □

Notice that bisimulations for relation-changing modal logics relate current states and current accessibility relations of the models. Depending on which operator we are considering, different zig/zag conditions are added. Zig and zag for $\mathcal{ML}$-bisimulations are the correspondent conditions to capture $\Diamond$: they talk about the successors of the current state. Conditions for relation-changing modal logics are the same, but also keeping track of the modifications already done, and changing the relation according to the semantics of the operators. For instance, $\langle \mathsf{sb} \rangle$-zig/zag establish that there are successors of the current states that are related, and delete the edges that connect them. For $\langle \mathsf{sw} \rangle$ is the same but swapping edges instead deleting. Conditions for $\langle \mathsf{br} \rangle$ require that there exist unreachable points from the current states, and put edges to them in the accessibility relation. Global cases are similar to local cases, but without changing the current state.

As we mentioned at the beginning of this chapter, we can define model equivalence in terms of two-player games, where Spoiler and Duplicator make explicit the modifications on the structure.

Definition 4.2.3 (Ehrenfeucht-Fraïssé Games). *Let $\mathcal{M}, w$ and $\mathcal{M}', w'$ be two pointed models and let $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{gbr} \rangle\}$.*

An Ehrenfeucht-Fraïssé game *$EF_{\blacklozenge}(\mathcal{M}, \mathcal{M}', w, w')$ is defined as follows. There are two players called* Spoiler *and* Duplicator. *Duplicator immediately loses if w and w' 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. If $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{sw} \rangle\}$, the game continues in one of the following ways:*

1. *Spoiler chooses v, a successor of w in the current model. If w has no successors, then Duplicator wins. Duplicator has to choose v', a successor of w' in the current model, such that v and v' agree. If there is no such successor, Spoiler wins. Otherwise the game continues with $EF_{\blacklozenge}(\mathcal{M}, \mathcal{M}', v, v')$. If Spoiler starts by choosing a successor v' of w', the same process has to be followed by exchanging the models where each player has to choose.*

2. *If $\blacklozenge = \langle \mathsf{sb} \rangle$, Spoiler chooses v, a successor of w and deletes the edge wv. If w has no successors, then Duplicator wins. Otherwise, Duplicator has to choose v', a successor of w' in the current model, such that v and v' agree and deletes the edge $w'v'$. If there is no such successor, Spoiler wins. Otherwise the game continues with $EF_{\blacklozenge}(\mathcal{M}^-_{wv}, \mathcal{M}'^-_{w'v'}, v, v')$. If Spoiler starts by choosing a successor v' of w', the same process has to be followed by exchanging the models where each player has to choose.*

3. *If $\blacklozenge = \langle \mathsf{sw} \rangle$, Spoiler chooses v, a successor of w and swaps around the edge wv. If w has no successors, then Duplicator wins. Otherwise, Duplicator has to choose v', a successor of w' in the current model, such that v and v' agree and swaps around the edge $w'v'$. If there is no such successor, Spoiler wins. Otherwise the game continues with $EF_{\blacklozenge}(\mathcal{M}^*_{wv}, \mathcal{M}'^*_{w'v'}, v, v')$. If Spoiler starts by choosing a successor v' of w', the same process has to be followed by exchanging the models where each player has to choose.*

If $\blacklozenge = \langle \mathsf{br} \rangle$, the game continues in one of the following ways:

1. *If Spoiler chooses v, a successor of w in the current model, then Duplicator has to choose v', a successor of w' in the current model, such that v and v' agree. If there is no such*

successor, Spoiler wins. The game continues with $EF_{\blacklozenge}(\mathcal{M}, \mathcal{M}', v, v')$. If Spoiler starts by choosing a successor v' of w', the same process has to be followed by exchanging the models where each player has to choose.

2. *If Spoiler chooses v, an element that is not a successor of w and adds the edge wv, then Duplicator has to choose v', a non-successor of w' in the current model, such that v and v' agree and adds the edge $w'v'$. The game continues with $EF_{\blacklozenge}(\mathcal{M}^{+}_{wv}, \mathcal{M}'^{+}_{w'v'}, v, v')$. If Spoiler starts by choosing a successor v' of w', the same process has to be followed by exchanging the models where each player has to choose.*

If the global counterparts are considered, the previous rules can be applied by choosing u, v, u', v' arbitrary elements of the models, instead of starting from the evaluation point. In such case, the game continues with $EF_{\blacklozenge}(\mathcal{M}^{\otimes}_{uv}, \mathcal{M}'^{\otimes}_{u'v'}, w, w')$, with $\otimes \in \{-, *, +\}$.

The winning conditions for the game $EF_{\blacklozenge}(\mathcal{M}, \mathcal{M}', w, w')$ are the same that for $\mathcal{ML}$. Given two pointed models $\mathcal{M}, w$ and $\mathcal{M}', w'$ we will write $\mathcal{M}, w \equiv^{EF}_{\blacklozenge} \mathcal{M}', w'$ when Duplicator has a winning strategy for $EF_{\blacklozenge}(\mathcal{M}, \mathcal{M}', w, w')$.

Bisimulations, games and Theorem 4.2.2 provide us with tools to investigate the expressivity of relation-changing modal logics. We can use these tools to compare the logics among them, and also with others. Definition 4.2.4 formalizes how we compare the expressive power of two logics.

Definition 4.2.4 ($\mathcal{L} \le \mathcal{L}'$). *We say that $\mathcal{L}'$ is* at least as expressive as *$\mathcal{L}$ (notation $\mathcal{L} \le \mathcal{L}'$) 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*

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

$\mathcal{M}$ is seen as a model of $\mathcal{L}$ on the left and as a model of $\mathcal{L}'$ on the right, and we use in each case the appropriate semantic relation $\models_{\mathcal{L}}$ or $\models_{\mathcal{L}'}$ as required.

We say that $\mathcal{L}$ and $\mathcal{L}'$ are incomparable *(notation $\mathcal{L} \ne \mathcal{L}'$) if $\mathcal{L} \not\le \mathcal{L}'$ and $\mathcal{L}' \not\le \mathcal{L}$.*

We say that $\mathcal{L}'$ is strictly more expressive *than $\mathcal{L}$ (notation $\mathcal{L} < \mathcal{L}'$) if $\mathcal{L} \le \mathcal{L}'$ but not $\mathcal{L}' \le \mathcal{L}$.*

The $\le$ relation indicates that we can embed one language into another. To do this, we need an equivalence preserving translation from the first language to the second one. Its strict version is $<$, that indicates that the second language can express strictly more than the first one. Incomparability relation says than any of the two languages cannot be embedded in the other, i.e., they are able to say different things. These definitions will be used in the next section, which is dedicated to compare the expressive power of relation-changing languages.

4.3 COMPARING LANGUAGES

We have introduced all the notions that we need to study the expressivity of a language in the previous section. But also we defined a way to relate different languages, according to what they are able to describe. This is the work in this section: we will take a pair of languages, and using the machinery introduced before we will be able to say if one of them is more (or equally) expressive than the other one, or if they are incomparable.

The first case, is to compare relation-changing modal logics with the basic modal logic. Their lack of tree and finite model property already tells us than they are more expressive than $\mathcal{ML}$, but in this section we provide a simpler proof of this result. The proof is direct from definitions: we have to find bisimilar models for $\mathcal{ML}$ than can be distinguished by the other logics, given that we can trivially embed $\mathcal{ML}$ in any of them.

Theorem 4.3.1. *$\mathcal{ML} < \mathcal{ML}(\blacklozenge)$, with $\blacklozenge \in \{\langle\mathsf{sb}\rangle, \langle\mathsf{gsb}\rangle, \langle\mathsf{sw}\rangle, \langle\mathsf{gsw}\rangle, \langle\mathsf{br}\rangle, \langle\mathsf{gbr}\rangle\}$.*

Proof. First, we have to provide a translation from $\mathcal{ML}$ formulas to $\mathcal{ML}(\blacklozenge)$. This is trivial, given that $\mathcal{ML}$ is a syntactic fragment of $\mathcal{ML}(\blacklozenge)$. To prove $\mathcal{ML}(\blacklozenge) \not\leq \mathcal{ML}$ we show two models that are bisimilar in $\mathcal{ML}$ and $\mathcal{ML}(\blacklozenge)$ formulas that distinguish them. Models in Figure 9 are $\mathcal{ML}$-bisimilar: w and w' agree propositionally (their valuations are empty), and in each model we can always find bisimilar successors. The bisimulation Z relates w with all the states of $\mathcal{M}'$.

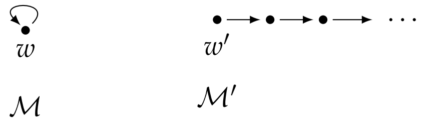


Figure 9: $\mathcal{ML}$-bisimilar models.

Now we have to find an $\mathcal{ML}(\blacklozenge)$-formula φ, with $\blacklozenge \in \{\langle\mathsf{sb}\rangle, \langle\mathsf{gsb}\rangle, \langle\mathsf{sw}\rangle, \langle\mathsf{gsw}\rangle, \langle\mathsf{br}\rangle, \langle\mathsf{gbr}\rangle\}$ such that φ holds in one of the models and does not hold in the other one. Formulas listed below hold at $\mathcal{M}', w'$ but do not hold at $\mathcal{M}, w$.

1. $\langle\mathsf{sb}\rangle\Diamond\top$. Clearly, after deleting one adjacent edge from w, we have not more successors, but we have it starting from w' (we have an infinite chain in $\mathcal{M}'$).
2. $\langle\mathsf{gsb}\rangle\Diamond\top$. Same as for the local case.
3. $\langle\mathsf{sw}\rangle\Diamond\Box\bot$. Swapping edges from w, we always keep the same model variant, and w always has a successor. Starting from w' by swapping an edge and returning to it after that, we reach a dead end.
4. $\langle\mathsf{gsw}\rangle\Box\bot$. For the global case we can do the same trick as the previous one.
5. $\langle\mathsf{br}\rangle\top$. There are not isolated parts of the model in the leftmost one, but in the rightmost model we have infinite possibilities to bridge.
6. $\langle\mathsf{gbr}\rangle\top$. We can do the same as for the local version of the operation.

□

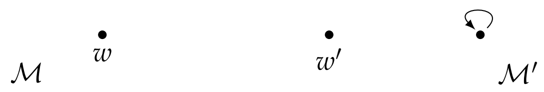
It was easy to see that $\mathcal{ML}$ is strictly less expressive than relation-changing modal logics. Dynamic behaviour obviously brings us unexpected consequences, such as we have seen in examples of Section 3.2, or the lack of model properties. The challenge now is to investigate which is the relation between the expressive power of the relation-changing modal logics. *"Can we embed sabotage in swap?". "Are local and global versions of the operators equally expressive?". "Which languages are incomparable?".* This is the kind of questions that we want and we are able to answer.

Most of the comparisons have been already studied in [Areces *et al.*, 2012], establishing that relation-changing modal logics are all incomparable among them. This is not a trivial result, given that we can expect some kind of connection at least between the local and the global versions of each operator, but as we will see in the next lemma, local or global effects allow us to describe different things. Some cases are easy to check, but there are others in which we need more complex structures to distinguish two languages.

Lemma 4.3.2. *For every pair of pointed models $\mathcal{M}, w$ and $\mathcal{M}', w'$ in Figure 10, and for all corresponding formulas φ of the column "Distinct by", we have $\mathcal{M}, w \not\models \varphi$ and $\mathcal{M}', w' \models \varphi$. Moreover, for all corresponding logics $\mathcal{L}$ of the column "Bisimilar for", we have that (w, R) and (w', R') are in an $\mathcal{L}$-bisimulation, where R and R' are the accessibility relations of $\mathcal{M}$ and $\mathcal{M}'$ respectively.*

Proof. That the pairs of models disagree on the given formulas can be easily verified. That the models are bisimilar for the given logics is also easily verified in most cases. Let see some of them in detail.

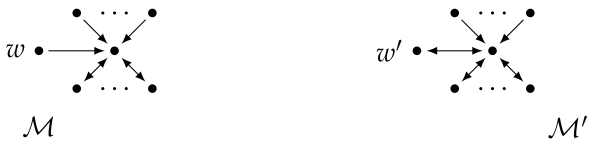
Consider the following models:



We will check the conditions to show that the two models are bisimilar for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ and for $\mathcal{ML}(\langle \mathsf{sw} \rangle)$. Clearly all the states agree propositionally (their valuations are empty). For zig and zag conditions, we need to check if both have bisimilar successors, which holds because there are not successors at all. The same happens with $\langle \mathsf{sb} \rangle$-zig/zag and $\langle \mathsf{sw} \rangle$-zig/zag: the lack of successors makes the conditions true. Now we can prove that $\mathcal{ML}(\blacklozenge) \not\leq \mathcal{ML}(\langle \mathsf{sb} \rangle)$ and $\mathcal{ML}(\blacklozenge) \not\leq \mathcal{ML}(\langle \mathsf{sw} \rangle)$, for $\blacklozenge \in \{\langle \mathsf{gsb} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$. We have to check now that there are formulas of such logics that distinguish the two models. The $\mathcal{ML}(\langle \mathsf{br} \rangle)$-formula $\langle \mathsf{br} \rangle \langle \mathsf{br} \rangle \top$ holds at $\mathcal{M}', w'$ but not at $\mathcal{M}, w$. Checking $\langle \mathsf{br} \rangle$-zag, it fails starting from w' and finding two states to reach with a new edge, while starting from w we can just reach one. The same happens with the $\langle \mathsf{gbr} \rangle$-formula $\langle \mathsf{gbr} \rangle \langle \mathsf{gbr} \rangle \top$, which allows us to add two edges in $\mathcal{M}'$ but this is not possible in $\mathcal{M}$. Formulas $\langle \mathsf{gsb} \rangle \top$ and $\langle \mathsf{gsw} \rangle \top$ also hold at $\mathcal{M}', w'$: it is possible to find in some part of the model an edge to delete or swap respectively, but not in the first model.

Now let us consider a more complicated case. The two pointed models below are $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$-bisimilar: they are the same graph with a different evaluation point. The graph is a star that has infinitely many ingoing branches, and infinitely many ingoing-outgoing branches. w is a point located at the end of an ingoing branch, and w' is at the end of an ingoing-outgoing branch.

To argue that the two models are bisimilar, it is easy to check the $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$-bisimulation as an Ehrenfeucht-Fraïssé game between Spoiler and Duplicator. If Spoiler moves to the center of the star, Duplicator can do the same and both situations



become indistinguishable. If Spoiler deletes one of the ingoing edges that has w or w' as origin, then Duplicator does the same on the other graph, and any further edge deletion can also be imitated. If Spoiler deletes the outgoing edge that goes from the center of the graph towards w', then Duplicator can delete any outgoing edge without changing the graph, given that there are infinitely many edges of both kinds. The $\mathcal{ML}(\langle \mathsf{sb} \rangle)$-formula $\langle \mathsf{sb} \rangle \Diamond \Box \bot$ holds at $\mathcal{M}', w'$, because it is possible to delete the edge when we move to the center of the star, and applying $\Diamond$ we move back to w' which now has no successors. In $\mathcal{M}$, it is not possible to move to a dead end after deleting an edge, then the formula does not hold.

The rest of the cases together with the previous are showed in Figure 10. In the third row, the given models are bisimilar for $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$ and $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ because they are bisimilar for $\mathcal{ML}$, they are acyclic and (for $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$) they contain the same number of edges. In the fourth row, both models are $\mathcal{ML}(\langle \mathsf{br} \rangle)$-bisimilar since they are infinite, hence one can add as many links as needed to points that are modally bisimilar.

$\Box$

The case $\mathcal{ML}(\langle \mathsf{gbr} \rangle) \not\leq \mathcal{ML}(\langle \mathsf{br} \rangle)$ remains to be checked. A complex construction is required for this case and we treat it separately in the following lemma.

Lemma 4.3.3. $\mathcal{ML}(\langle \mathsf{gbr} \rangle) \not\leq \mathcal{ML}(\langle \mathsf{br} \rangle)$.

Proof. We give the description of an infinite model $\mathcal{M}$ with two states w and v such that $\mathcal{M}, w$ and $\mathcal{M}, v$ are $\mathcal{ML}(\langle \mathsf{br} \rangle)$-bisimilar and there is an $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$-formula φ such that $\mathcal{M}, w \models \varphi$ and $\mathcal{M}, v \not\models \varphi$.

Let a *piece* be a part of a model with a finite and non-zero number of states and some relation between them. Let a *collection* be the disjoint union of an infinite number of copies of the same piece. Let $\mathcal{M}$ be the disjoint union of all collections obtained from all possible pieces.

Let w be a state of a piece of the following form: $\bullet_w \rightarrow \bullet$, and v a state of a piece of the following form: $\bullet \leftarrow \bullet_v \rightarrow \bullet$.

First, notice that $\langle \mathsf{gbr} \rangle \Box \Diamond \top$ is true at $\mathcal{M}, w$ and false at $\mathcal{M}, v$. We will check $\mathcal{ML}(\langle \mathsf{br} \rangle)$-bisimilarity by using Ehrenfeucht-Fraïsse games and presenting a winning strategy for Duplicator.

$\mathcal{M}, w$ and $\mathcal{M}, v$ are bisimilar for the basic modal logic. We show that after any $\langle \mathsf{br} \rangle$-move in any model, it is possible to do a $\langle \mathsf{br} \rangle$-move in the other model that leads to a modally bisimilar part of the model. Assume Spoiler does a bridge to some part of the model. As the model is made of pieces of finite size, the evaluation point moves to a finite connected component. Duplicator only needs to do a bridge to a finite connected component of the same shape. $\Box$

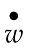
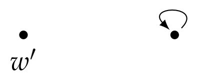
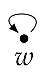
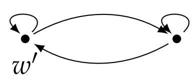
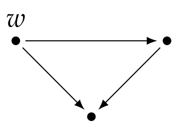
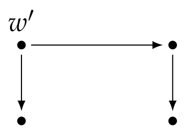
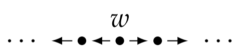
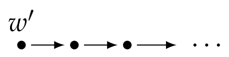
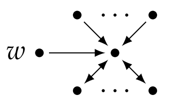
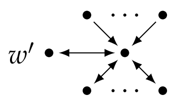
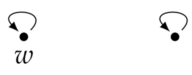
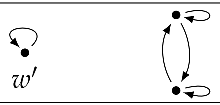
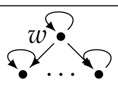
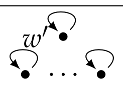
$\mathcal{M}$	$\mathcal{M}'$	Distinct by	Bisimilar for
		$\langle\mathsf{br}\rangle\langle\mathsf{br}\rangle\top$ $\langle\mathsf{gbr}\rangle\langle\mathsf{gbr}\rangle\top$ $\langle\mathsf{gsb}\rangle\top$ $\langle\mathsf{gsw}\rangle\top$	$\mathcal{ML}(\langle\mathsf{sb}\rangle)$ $\mathcal{ML}(\langle\mathsf{sw}\rangle)$
		$\langle\mathsf{sb}\rangle\Diamond\top$ $\langle\mathsf{gsb}\rangle\Diamond\top$	$\mathcal{ML}(\langle\mathsf{sw}\rangle)$ $\mathcal{ML}(\langle\mathsf{br}\rangle)$ $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$ $\mathcal{ML}(\langle\mathsf{gbr}\rangle)$
		$\langle\mathsf{sw}\rangle\Diamond\Diamond\Diamond\Box\bot$ $\Diamond\langle\mathsf{gsw}\rangle\Diamond\Diamond\Diamond\Box\bot$ $\langle\mathsf{br}\rangle\langle\mathsf{br}\rangle\top$ $\langle\mathsf{gbr}\rangle^6\langle\mathsf{gbr}\rangle\top$	$\mathcal{ML}(\langle\mathsf{gsb}\rangle)$ $\mathcal{ML}(\langle\mathsf{sb}\rangle)$
		$\langle\mathsf{sw}\rangle\Diamond\Box\bot$ $\langle\mathsf{gsw}\rangle\Box\bot$	$\mathcal{ML}(\langle\mathsf{br}\rangle)$ $\mathcal{ML}(\langle\mathsf{gbr}\rangle)$
		$\langle\mathsf{sb}\rangle\Diamond\Box\bot$	$\mathcal{ML}(\langle\mathsf{gsb}\rangle)$
		$\langle\mathsf{br}\rangle^3\top$ $\langle\mathsf{gbr}\rangle^3\top$	$\mathcal{ML}(\langle\mathsf{gsw}\rangle)$
		$\langle\mathsf{br}\rangle\top$	$\mathcal{ML}(\langle\mathsf{gbr}\rangle)$

Figure 10: Bisimilar models and distinguishing formulas.

Theorem 4.3.4. *For all* $\blacklozenge_1, \blacklozenge_2 \in \{\langle\mathsf{sb}\rangle, \langle\mathsf{gsb}\rangle, \langle\mathsf{sw}\rangle, \langle\mathsf{gsw}\rangle, \langle\mathsf{br}\rangle, \langle\mathsf{gbr}\rangle\}$ *with* $\blacklozenge_1 \neq \blacklozenge_2$, $\mathcal{ML}(\blacklozenge_1)$ *and* $\mathcal{ML}(\blacklozenge_2)$, $\mathcal{ML}(\blacklozenge_1) \neq \mathcal{ML}(\blacklozenge_2)$ *(they are incomparable), except for the comparison* $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$ *and* $\mathcal{ML}(\langle\mathsf{sw}\rangle)$.

Proof. Direct from Lemmas 4.3.2 and 4.3.3. ☐

The only comparison we have not obtained, is the case $\mathcal{ML}(\langle\mathsf{sw}\rangle) \not\leq \mathcal{ML}(\langle\mathsf{gsw}\rangle)$. The question is open, but we conjecture that they are also incomparable as for all the other logics.

Conjecture 4.3.5. $\mathcal{ML}(\langle\mathsf{sw}\rangle) \neq \mathcal{ML}(\langle\mathsf{gsw}\rangle)$.

Adding relation-changing operators to the basic modal logic increases its expressive power, but according to the results we just showed, each logic allows to express different things. In Chapter 3, we have already seen that lack of tree and finite model properties is proved by enforcing different structures for each relation-changing o-perator. In this chapter, we proved using standard tools such as bisimulations and

Ehrenfeucht-Fraïsse games that the expressive power of the six logics are all incomparable among them, except for the case $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ and $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ (we conjecture that they are also incomparable). In the following chapter we will start to investigate in detail the computational behaviour of relation-changing modal logics.

4.4 A MORE GENERAL PERSPECTIVE

Some work related to the expressive power of the languages introduced in this thesis remains to be done. We defined a notion of bisimulation which captures exactly the behaviour of relation-changing operators. For each language, we define bisimulation as a relation linking states and the updated accessibility relation of the models, and we proved that two bisimilar models satisfy the same formulas of the corresponding language. We have specific conditions depending of the operators we are considering, but the conditions are defined uniformly. It is possible to define a more general notion of bisimulation which instead of considering particular cases of updates (deleting, swapping or adding edges) might consider update functions in general. It would be easy to show that under certain circumstances, we can use the same ideas as for sabotage, swap and bridge operators to define a proper notion of bisimulation for each model modifier.

We discussed general results about bisimulation, but the same ideas can be also applied to particular cases. Other modal modifiers have been investigated, for instance, in Dynamic Epistemic Logic. In some cases, the languages do not increase the expressive power of $\mathcal{ML}$: they can be translated to $\mathcal{ML}$ via *reduction axioms*. As a consequence, the notion of bisimulation is the same as for $\mathcal{ML}$. However, there are other cases in which the changes produced by the operator cannot be expressed in $\mathcal{ML}$. For these languages, we conjecture that we can apply the general techniques as for relation-changing logics to get a proper notion of bisimulation.

Another interesting problem to investigate is the *van Benthem Characterization* [van Benthem, 1977]. The original Characterization Theorem establishes that a $\mathcal{FOL}$-formula φ is equivalent to the translation of an $\mathcal{ML}$-formula if and only if φ is invariant under bisimulations. In [Areces *et al.*, 2013a] it is shown that van Benthem characterization theorems hold for any modal language which satisfies certain adequacy conditions. It would be interesting to check these adequacy conditions and investigate characterization for relation-changing modal logics.

5

The Satisfiability Problem

The Entscheidungsproblem is solved when we know a procedure that allows for any given logical expression to decide by finitely many operations its validity or satisfiability. (...) The Entscheidungsproblem must be considered the main problem of mathematical logic.

from "Grundzüge der theoretischen Logik", David Hilbert and Wilhelm Ackerman.

A classical problem that has to be investigated when we study logic is *the satisfiability problem*, i.e., given any formula of the language decide if there is a model which satisfies the formula. In this chapter we investigate the satisfiability problem for the local versions of the relation-changing modal logics we introduced. As we showed in the previous chapter, relation-changing operators increase the expressive power of $\mathcal{ML}$, but we do not know yet how much. In [Areces *et al.*, 2013b] we already proved that the satisfiability problem for $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ is undecidable. We follow the same ideas to get undecidability for the other relation-changing modal logics. We also showed the lack of the tree and the finite model property for the six logics, which leaves us in the border of undecidability. We are indeed going to see that we can reduce the undecidable satisfiability problem of $\mathcal{ML}(\circledR, \circledk)$ to the one of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, $\mathcal{ML}(\langle \mathsf{br} \rangle)$ and $\mathcal{ML}(\langle \mathsf{sw} \rangle)$.

First, we translate memory logics models to standard Kripke models. The idea of the translation is that it forces some constraint in the shape of the models in which we will evaluate the formulas. The intended models, follow the structure defined for infinite models of Section 3.3. Then, we translate $\mathcal{ML}(\circledR, \circledk)$-formulas to relation-changing formulas. In the next sections we introduce the translations for the local versions of sabotage, bridge and swap respectively. The behaviour of swap is quite different and harder to capture, as we will see in other chapters, constructions for swap are always harder than for the other operators.

5.1 Sabotage Logic

We start by proving that the satisfiability problem for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ is undecidable. First, we provide a translation from formulas of the memory logic $\mathcal{ML}(\circledR, \circledk)$ to $\mathcal{ML}(\langle \mathsf{sb} \rangle)$-formulas. In order to simulate the behaviour of the operators $\circledR$ and $\circledk$ without having a memory in the model, we impose constraints on the models where we evaluate the translated formula. Then we prove that a $\mathcal{ML}(\circledR, \circledk)$-formula is satisfiable if and only if, the translation of such a formula (in addition to the constraints we define) is satisfiable.

Definition 5.1.1. *Let $s \in$ PROP, we define Conds as the conjunction of the following formulas:*

$$\begin{array}{ll}
(1) & s \wedge \Box\neg s \wedge \Box\Diamond s \\
(2) & \Box\Box(s \rightarrow \neg\Diamond s) \\
(3) & [\mathsf{sb}][\mathsf{sb}](s \rightarrow \Box\Diamond s) \\
(4) & \Box[\mathsf{sb}](s \rightarrow \Diamond\neg\Diamond s) \\
(5) & \Box\Box(\neg s \rightarrow \Diamond(s \wedge \neg\Diamond s)) \\
(6) & \Box[\mathsf{sb}](\neg s \rightarrow [\mathsf{sb}](s \rightarrow \Box\Box(\neg s \rightarrow \Diamond s))) \\
(7) & \Box\Box(\neg s \rightarrow [\mathsf{sb}](s \rightarrow \Diamond\Diamond(\neg s \wedge \neg\Diamond s))) \\
(Spy) & \Box\Box(\neg s \rightarrow [\mathsf{sb}](s \rightarrow \Diamond\neg\Diamond s)).
\end{array}$$

Let us call s (for *spy point*) a node satisfying *Conds* in an arbitrary model. Then, the point s satisfies the propositional symbol s, and is related with all the states of the connected component of the model in the two directions. Formula (1) ensures that the propositional symbol s is satisfied at the evaluation point, and is not satisfied in any successor. As a consequence, the state s is irreflexive. It also says that all the successors can see an s-state.

(2) ensures that all the s-states that are accessible in two steps from the evaluation point, have no successors satisfying s.

(3) ensures that after deleting two edges and reaching an s-state, the property that all the successors can see an s-state is maintained.

The formula (4) establishes that for all the successors, after deleting an edge and reaching an s-state, there is a successor which cannot see any s-state (it was the only successor satisfying s).

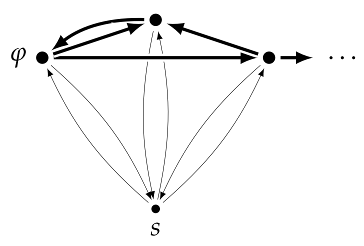
(5) says that reaching some state in two steps that does not satisfy s, there is always an s-state which is reachable and has no successors satisfying s.

(6) ensures that after eliminating the edge from a $\neg s$-state (which is no longer accessible from the evaluation point in two steps) to an s-state, the remaining $\neg s$-states still have an edge pointing to some state satisfying s.

(7) ensures that all the states reachable in two steps (which do not satisfy s) have only one successor labeled by s.

(Spy) establishes that states that are accessible in two steps are also accessible in one step.

Next, we will see an example showing how we will use *Conds*. The idea is to pick an $\mathcal{ML}(ⓡ, ⓚ)$ model, and add a spy point to satisfy *Conds*. A model where $\mathcal{M}, s \models Conds$ is illustrated below:



In this picture, the thick points and lines represent the model of the initial memory logic formula that can be extracted from the whole model. We introduce some properties of the models satisfying *Conds*, that will be useful in the equisatisfiability proof.

Proposition 5.1.2. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, $w \in W$. If $\mathcal{M}, w \models Conds$, then the following properties hold:*

1. *w is the only state in $\mathcal{M}$ that satisfies s in the connected component generated by w.*
2. *For all states $v \in W$ such that $v \neq w$, we have that if $(w, v) \in R$ then $(v, w) \in R$, and if $(w, v) \in R^*$ then $(w, v) \in R$ (w is a spy point).*

Proposition 5.1.2 enumerates the main properties of the spy point: it is the only spy point in the connected component, and each time that there is an outgoing edge to some state of the model, there is also an edge coming back.

Now we introduce the translation from $\mathcal{ML}(ⓡ, ⓚ)$-formulas to $\mathcal{ML}(\langle \mathsf{sb}\rangle)$-formulas.

Definition 5.1.3. *Let φ be an $\mathcal{ML}(ⓡ, ⓚ)$-formula that does not contain the propositional symbol s. We define $\mathsf{Tr}(\varphi) = \Diamond(\varphi)'$, where $(\ )'$ is defined as follows:*

$$\begin{array}{lcl}
(p)' & = & p \quad \textit{for } p \in \textit{PROP appearing in } \varphi \\
(ⓚ)' & = & \neg\Diamond s \\
(\neg\psi)' & = & \neg(\psi)' \\
(\psi \wedge \chi)' & = & (\psi)' \wedge (\chi)' \\
(\Diamond\psi)' & = & \Diamond(\neg s \wedge (\psi)') \\
(ⓡ\psi)' & = & (\Diamond s \to \langle \mathsf{sb}\rangle(s \wedge \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\psi)'))) \ \wedge \ (\neg\Diamond s \to (\psi)').
\end{array}$$

Boolean and modal cases are obvious. ⓡ is represented by removing the edges from the spy point to the state we want to memorize and from this state to the spy point. Notice how the translation behaves: if the point has already been memorized ($\neg\Diamond s$), then nothing needs to be done and translation continues; otherwise ($\Diamond s$), we make s inaccessible using $\langle \mathsf{sb}\rangle$ and we also delete the arrow from s to the current point. ⓚ is represented by checking whether there is an edge pointing to the spy point or not.

Theorem 5.1.4. *Let φ be a formula of $\mathcal{ML}(ⓡ, ⓚ)$ that does not contain the propositional symbol s. Then, φ and $\mathsf{Tr}(\varphi) \wedge Conds$ are equisatisfiable.*

Proof. We will prove that φ is satisfiable if and only if $\mathsf{Tr}(\varphi) \wedge Conds$ is satisfiable.

($\Leftarrow$) Suppose that $\mathsf{Tr}(\varphi) \wedge Conds$ is satisfiable, i.e., there exists a model $\mathcal{M} = \langle W, R, V\rangle$, and $s \in W$ such that $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$ and $\langle W, R, V\rangle, s \models Conds$. Then we can define the model $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$ where

$$\begin{array}{ll}
W' & = \{v \mid (s, v) \in R\} \\
R' & = R \cap (W' \times W') \\
V'(p) & = V(p) \cap W' \quad \text{for } p \in \mathsf{PROP}.
\end{array}$$

Let $w' \in W'$ be a state s.t. $(s, w') \in R$ and $\langle W, R, V\rangle, w' \models (\varphi)'$ (because $\langle W, R, V\rangle, s \models \Diamond(\varphi)'$). We will prove

$$\langle W', R', V', M'\rangle, v \models \psi \ \text{ iff } \ \langle W, R(M'), V\rangle, v \models (\psi)',$$

where $v \in W'$, $M' \subseteq W'$, $\psi \in \mathsf{FORM}$, and $R(M') = R \backslash \{(s, t), (t, s) \mid t \in M'\}$. In particular, when $M' = \emptyset$ we have that $\langle W', R', V', \emptyset\rangle, w' \models \varphi$ iff $\langle W, R, V\rangle, w' \models (\varphi)'$.

Then we do structural induction on ψ. We have two base cases:

$\boldsymbol{\psi = p}$: Suppose that $\langle W', R', V', M'\rangle, v \models p$. By definition of $\models$, $v \in V'(p)$, and this is equivalent to $v \in V(p) \cap W'$ by definition of V'. Because $v \in V(p)$, by $\models$ we have $\langle W, R(M'), V\rangle, v \models p$, and by definition of $(\ )'$ this is equivalent to $\langle W, R(M'), V\rangle, v \models (p)'$.

$\boldsymbol{\psi} =$ ⓚ: Suppose that $\langle W', R', V', M'\rangle, v \models$ ⓚ. By $\models$ we have $v \in M'$, and by Proposition 5.1.2 and definition of $R(M')$ we have $(v, s) \notin R(M')$ and $\langle W, R(M'), V\rangle, s \models s$. Then by $\models$ $\langle W, R(M'), V\rangle, v \models \neg\Diamond s$, and by definition of $(\ )'$ this is equivalent to $\langle W, R(M'), V\rangle, v \models$ (ⓚ)$'$.

Now we prove inductive cases.

$\boldsymbol{\psi = \neg\phi}$: Suppose $\langle W', R', V', M'\rangle, v \models \neg\phi$. By definition of $\models$, $\langle W', R', V', M'\rangle, v \not\models \phi$. By I.H., we have $\langle W, R(M'), V\rangle, v \not\models (\phi)'$, iff $\langle W, R(M'), V\rangle, v \models \neg(\phi)'$. Then, by definition of $(\ )'$, $\langle W, R(M'), V\rangle, v \models (\neg\phi)'$.

$\boldsymbol{\psi = \phi \wedge \chi}$: Suppose $\langle W', R', V', M'\rangle, v \models \phi \wedge \chi$. By $\models$, $\langle W', R', V', M'\rangle, v \models \phi$ and $\langle W', R', V', M\rangle, v \models \chi$. By I.H., $\langle W, R(M'), V\rangle, v \models (\phi)'$ and $\langle W, R(M'), V\rangle, v \models (\chi)'$. Then we have $\langle W, R(M'), V\rangle, v \models (\phi)' \wedge (\chi)'$. Then by definition of $(\ )'$, $\langle W, R(M'), V\rangle, v \models (\phi \wedge \chi)'$.

$\boldsymbol{\psi = \Diamond\phi}$: Suppose $\langle W, R(M'), V\rangle, v \models (\Diamond\phi)'$. By definition of $(\ )'$, $\langle W, R(M'), V\rangle, v \models \Diamond(\neg s \wedge (\phi)')$. By $\models$, there is $v' \in W$ s.t. $(v, v') \in R(M')$ and $\langle W, R(M'), V\rangle, v' \models \neg s \wedge (\phi)'$. Then we have $\langle W, R(M'), V\rangle, v' \models \neg s$ and $\langle W, R(M'), V\rangle, v' \models (\phi)'$. By I.H., $\langle W', R', V', M'\rangle, v' \models \phi$, hence by $\models$ and Proposition 5.1.2, we have $\langle W', R', V', M'\rangle, v \models \Diamond\phi$.

$\boldsymbol{\psi} =$ ⓡ$\boldsymbol{\phi}$: Suppose $\langle W, R(M'), V\rangle, v \models$ (ⓡ$\phi)'$. By definition of $(\ )'$ and $\models$,

$$\langle W, R(M'), V\rangle, v \models \Diamond s \to \langle \mathsf{sb}\rangle(s \wedge \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\phi)')) \text{ and}$$
$$\langle W, R(M'), V\rangle, v \models \neg\Diamond s \to (\phi)'.$$

We will prove each conjunct separately. First, suppose $\langle W, R(M'), V\rangle, v \models \Diamond s$. Then $(v, s) \in R(M')$ (by Proposition 5.1.2). We want to prove that $\langle W, R(M'), V\rangle, v \models \langle \mathsf{sb}\rangle(s \wedge \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\phi)'))$. By assumption we know $(v, s) \in R(M')$ then $(s, v) \in R(M')$, because in $R(M')$ we always delete pairs in the two directions and by Proposition 5.1.2. Then we only need to prove that $\langle W, R(M')^-_{vs}, V\rangle, s \models s \wedge \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\phi)')$. It is trivial that $\langle W, (R(M'))^-_{vs}, V\rangle, s \models s$. Let us see that $\langle W, (R(M'))^-_{vs}, V\rangle, s \models \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\phi)')$. Because $(s, v) \in (R(M'))^-_{vs}$, it suffices to prove $\langle W, R(M')^-_{vs,sv}, V\rangle, v \models \neg\Diamond s \wedge (\phi)'$. First conjunct is trivial because $(v, s) \notin R(M')^-_{vs,sv}$.

On the other hand, we know that for all t, $(t, s) \notin R(M')$ iff $(s, t) \notin R(M')$. Then by I.H., $\langle W, R(M')^-_{vs,sv}, V\rangle, v \models (\phi)'$ iff $\langle W', R', V', M \cup \{v\}\rangle, v \models \phi$. Hence, by $\models$ we have $\langle W', R', V', M'\rangle, v \models$ ⓡϕ.

Now suppose the other case, $\langle W, R(M'), V\rangle, v \models \neg\Diamond s$. By Proposition 5.1.2, we know $(v, s) \notin R(M')$. By definition of $R(M')$, we have $(s, v) \notin R(M')$. Then $v \in M'$, and by I.H. we have $\langle W', R', V', M'\rangle, v \models$ ⓡϕ.

($\Rightarrow$) Suppose that φ is satisfiable, i.e., there exists a model $\mathcal{M} = \langle W, R, V, \emptyset \rangle$ and $w \in W$ such that $\langle W, R, V, \emptyset \rangle, w \models \varphi$.

Let s be a state that does not belong to W. Then we can define the model $\mathcal{M}' = \langle W', R', V' \rangle$ as follows:

$$\begin{array}{ll} W' & = W \cup \{s\} \\ R' & = R \cup \{(s,w) \mid w \in W\} \cup \{(w,s) \mid w \in W\} \\ V'(p) & = V(p) \quad \text{for } p \in \mathsf{PROP} \text{ appearing in } \varphi \\ V'(s) & = \{s\}. \end{array}$$

By construction of $\mathcal{M}'$, it is easy to check that $\mathcal{M}', s \models \mathit{Conds}$. Then we can verify that

$$\langle W, R, V, M \rangle, w \models \varphi \ \text{ iff } \ \langle W', R'(M), V' \rangle, s \models \mathsf{Tr}(\varphi),$$

where $R'(M)$ is defined as for the ($\Leftarrow$) direction of the proof.

Again we need to do structural induction. Boolean cases are easy, and it is also the case for ⓚ. If $\langle W, R, V, M \rangle, w \models \Diamond\psi$, then by construction of $\mathcal{M}'$ it is clear that $w \notin V'(s)$ and $\langle W', R'(M), V' \rangle, v \models (\psi)'$. If $\langle W, R, V, M \rangle, w \models$ ⓡψ, we can delete the edges (w, s) and (s, w) to simulate the storing of w in the memory (if those pairs are not in R' means $w \in M$) and continue by evaluating the rest of the translation $(\ )'$ (steps are similar than for the ($\Leftarrow$) direction of the proof). □

From the previous theorem, we immediately get:

Theorem 5.1.5. *The satisfiability problem of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ is undecidable.*

We have proved that we can reduce the satisfiability problem of $\mathcal{ML}$(ⓡ, ⓚ) to the same problem for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$. Based on these results, and given that we can enforce infinite models with $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$-formulas, we conjecture that similar constructions can be done for the global version of sabotage, leaving it as future work.

5.2 BRIDGE LOGIC

In this section, we will reduce the satisfiability problem of $\mathcal{ML}$(ⓡ, ⓚ) to the satisfiability problem of $\mathcal{ML}(\langle \mathsf{br} \rangle)$. The idea is to simulate the behaviour of ⓡ by keeping a "spy point" and adding back and forth edges from the state to be remembered to the spy point. Then, the behaviour of ⓚ is captured by consulting if the current state points to the spy point. In this case, we will allow multiple spy points which are unconnected with the rest of the model, but form a connected component among them. However, the constraints we impose enforce that we always use the same spy point. Another difference with $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ is that we use a second set of states (called "bridge points") between spy points and the rest of the model, also unconnected from the rest, but connected among them. Because spy points are isolated, we need to add new edges to reach the rest of the model. If we directly add an edge to an arbitrary state, when we start the evaluation of a formula, we are introducing some undesired information (remember that new edges stand for visited states). For that reason we use bridge points to start the evaluation without introducing unwanted edges.

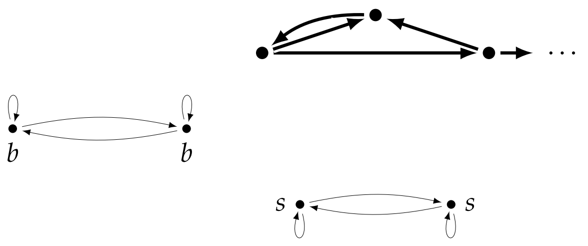
Next we introduce the conditions to enforce the intended models.

Definition 5.2.1. *Let $s, b \in \mathsf{PROP}$, we define Conds as the conjunction of the following formulas:*

$$\begin{array}{ll}
(1) & s \wedge [\mathsf{br}]\neg s \\
(s-connex) & \Box[\mathsf{br}]\neg s \\
(3) & \Box\Box s \\
(4) & \Box[\mathsf{br}]\Box\neg s \\
(5) & \Box\neg b \\
(6) & \langle\mathsf{br}\rangle(b \wedge [\mathsf{br}]\neg b \\
 & \wedge \Box[\mathsf{br}]\neg b \\
 & \wedge \Box\Box b \\
 & \wedge \Box[\mathsf{br}]\Box\neg b \\
 & \wedge \Box\neg s).
\end{array}$$

(1) establishes that the evaluation point satisfies s and it is reflexive. $(s - connex)$ ensures there is no unconnected s-state. (3) ensures that all the successors have only s-successors. (4) says that there are no edges from $\neg s$-states incoming to the connected component satisfying s. (5) ensures that no successor of s satisfies b. Finally, the formula (6) establishes that there is a totally connected component of states satisfying b (described exactly as for s) which is unreachable from the s-component.

The idea is to pick an $\mathcal{ML}(ⓡ, ⓚ)$ model, and add a cloud of spy points and a cloud of bridge points to satisfy *Conds*. A model where $\mathcal{M}, s \models Conds$ is illustrated below:



In this picture, the thick points and lines represent the model of the initial memory logic formula that can be extracted from the whole model.

The next proposition spells out the shape of the models we want to enforce.

Proposition 5.2.2. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, $w \in W$. If $\mathcal{M}, w \models Conds$, then the following properties hold:*

1. *$w \in V(s)$ and $V(s)$ is totally connected (for all $v, u \in V(s)$, we have $(v, u) \in R$ and $(u, v) \in R$).*
2. *If $w \in V(s)$ and $v \notin V(s)$ then $(w, v) \notin R$ and $(v, w) \notin R$.*
3. *$w \notin V(b)$ and $V(b)$ is totally connected (for all $v, u \in V(b)$, we have $(v, u) \in R$ and $(u, v) \in R$).*
4. *If $w \notin V(b)$ and $v \in V(b)$ then $(w, v) \notin R$ and $(v, w) \notin R$.*

Now we define the translation from $\mathcal{ML}(ⓡ, ⓚ)$-formulas to $\mathcal{ML}(\langle\mathsf{br}\rangle)$-formulas.

Definition 5.2.3. *Let φ be an $\mathcal{ML}(ⓡ,ⓚ)$-formula that does not contain the propositional symbols s and b. We define $\mathsf{Tr}(\varphi) = \langle \mathsf{br}\rangle(b \wedge \langle \mathsf{br}\rangle(\neg s \wedge (\varphi)'))$, where $(\ )'$ is defined as follows:*

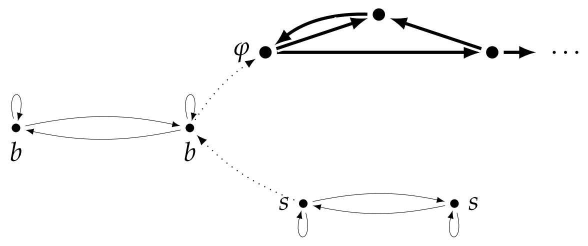
$$\begin{array}{lcl}
(p)' & = & p \quad \text{for } p \in PROP \text{ appearing in } \varphi \\
(ⓚ)' & = & \Diamond s \\
(\neg\psi)' & = & \neg(\psi)' \\
(\psi \wedge \chi)' & = & (\psi)' \wedge (\chi)' \\
(\Diamond\psi)' & = & \Diamond(\neg s \wedge \neg b \wedge (\psi)') \\
(ⓡ\psi)' & = & (\neg\Diamond s \rightarrow \langle \mathsf{br}\rangle(s \wedge \Diamond\neg s \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\psi)'))\ \wedge\ (\Diamond s \rightarrow (\psi)').
\end{array}$$

Boolean cases are obvious. $\Diamond\psi$ is satisfied if there is a successor where ψ holds, and we also check that this successor does not satisfies s and b. ⓡ is represented by adding an edge to the state to be remembered and some s-state in the two directions. In this way we identify visited states by checking if there is an edge to an s-state.

Then we can state:

Theorem 5.2.4. *Let φ be a formula of $\mathcal{ML}(ⓡ,ⓚ)$ that does not contain the propositional symbol s and b. Then, φ and $\mathsf{Tr}(\varphi) \wedge Conds$ are equisatisfiable.*

A model where $\mathcal{M}, s \models Conds \wedge Tr(\varphi)$ is illustrated below:



Proof. We will prove that φ is satisfiable if and only if $\mathsf{Tr}(\varphi) \wedge Conds$ is satisfiable.

($\Leftarrow$) Suppose that $\mathsf{Tr}(\varphi) \wedge Conds$ is satisfiable, i.e., there exists a model $\mathcal{M} = \langle W, R, V\rangle$, and $s \in W$ such that $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$ and $\langle W, R, V\rangle, s \models Conds$. Then we can define the model $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$, where

$$\begin{array}{ll}
W' & = \{v \mid (s, v) \notin R\} \\
R' & = R \cap (W' \times W') \\
V'(p) & = V(p) \cap W' \quad \text{for } p \in \mathsf{PROP}.
\end{array}$$

Let $w' \in W'$ be a state such that there is a $w'' \in W$ s.t $(s, w'') \in R$ and $(w'', w') \in R$ and $\langle W, R, V\rangle, w' \models (\varphi)'$ (because $\langle W, R, V\rangle, s \models \langle \mathsf{br}\rangle(b \wedge \langle \mathsf{br}\rangle(\neg s \wedge (\varphi)')))$. We will prove

$$\langle W', R', V', M'\rangle, v \models \psi \text{ iff } \langle W, R(M'), V\rangle, v \models (\psi)'$$

where $v \in W'$, $M' \subseteq W'$, $\psi \in \mathsf{FORM}$, and $R(M') = R \cup \{(s,t), (t,s) \mid t \in M'\}$. In particular, when $M' = \emptyset$ we have that $\langle W', R', V', \emptyset\rangle, w' \models \varphi$ iff $\langle W, R, V\rangle, w' \models (\varphi)'$.

Then we do structural induction on ψ. We have two base cases:

$\boldsymbol{\psi = p}$: Suppose that $\langle W', R', V', M' \rangle, v \models p$. By $\models$ we have $v \in V'(p)$, and this is equivalent to $v \in V(p) \cap W'$ by definition of V'. Because $v \in V(p)$, by $\models$ we have $\langle W, R(M'), V \rangle, v \models p$, and by definition of $(\ )'$ this is equivalent to $\langle W, R(M'), V \rangle, v \models (p)'$.

$\boldsymbol{\psi} = \textcircled{k}$: Suppose that $\langle W', R', V', M' \rangle, v \models \textcircled{k}$. By $\models$ we have $v \in M'$, and by Proposition 5.2.2 and definition of $R(M')$ we have $(v, s) \in R(M')$ and $\langle W, R(M'), V \rangle, s \models s$. Then by $\models$ $\langle W, R(M'), V \rangle, v \models \Diamond s$, and by definition of $(\ )'$ this is equivalent to $\langle W, R(M'), V \rangle, v \models (\textcircled{k})'$.

Now we prove inductive cases.

$\boldsymbol{\psi = \neg\phi}$ and $\boldsymbol{\psi = \phi \wedge \chi}$: follow by I.H.

$\boldsymbol{\psi = \Diamond\phi}$: Suppose $\langle W, R(M'), V \rangle, v \models (\Diamond\phi)'$. By definition of $(\ )'$ we have $\langle W, R(M'), V \rangle, v \models \Diamond(\neg s \wedge \neg b \wedge (\phi)')$. By $\models$, there exists $v' \in W$ s.t. $(v, v') \in R(M')$ and $\langle W, R(M'), V \rangle, v' \models \neg s \wedge \neg b \wedge (\phi)'$. Then (by $\models$) $\langle W, R(M'), V \rangle, v' \models \neg s$, $\langle W, R(M'), V \rangle, v' \models \neg b$ and $\langle W, R(M'), V \rangle, v' \models (\phi)'$. By I.H. we have $\langle W', R', V', M' \rangle, v' \models \phi$, hence by $\models$ and Proposition 5.2.2, $\langle W', R', V', M' \rangle, v \models \Diamond\phi$.

$\boldsymbol{\psi} = \textcircled{r}\boldsymbol{\phi}$: Suppose $\langle W, R(M'), V \rangle, v \models (\textcircled{r}\phi)'$. By definition of $(\ )'$ and $\models$,

$$\langle W, R(M'), V \rangle, v \models \neg\Diamond s \to \langle \mathsf{br} \rangle (s \wedge \Diamond\neg s \wedge \langle \mathsf{br} \rangle (\Diamond s \wedge (\phi)')) \text{ and}$$
$$\langle W, R(M'), V \rangle, v \models \Diamond s \to (\phi)'.$$

We will prove each conjunct separately. First, suppose $\langle W, R(M'), V \rangle, v \models \neg\Diamond s$. Then $(v, s) \notin R(M')$ (by Proposition 5.2.2). We want to prove that $\langle W, R(M'), V \rangle, v \models \langle \mathsf{br} \rangle (s \wedge \Diamond\neg s \wedge \langle \mathsf{br} \rangle (\Diamond s \wedge (\phi)'))$. By assumption and Proposition 5.2.2, we know that if $(v, s) \notin R(M')$ then $(s, v) \notin R(M')$ (because in $R(M')$ we always add pairs in the two directions). Then we only need to prove $\langle W, R(M')^+_{vs}, V \rangle, s \models s \wedge \Diamond\neg s \wedge \langle \mathsf{br} \rangle (\Diamond s \wedge (\phi)')$. We know $s \in V(s)$, then it results obvious that $\langle W, (R(M'))^+_{vs}, V \rangle, s \models s$. Then $\langle W, (R(M'))^+_{vs}, V \rangle, s \models \Diamond\neg s$, because the evaluation of the translation started at s and adding an edge to a b-state (that does not satisfy s by Proposition 5.2.2). It remains to see that $\langle W, (R(M'))^+_{vs}, V \rangle, s \models \langle \mathsf{br} \rangle (\Diamond s \wedge (\phi)')$. Because $(s, v) \notin (R(M'))^+_{vs}$, it suffices to prove $\langle W, R(M')^+_{vs,sv}, V \rangle, v \models \Diamond s \wedge (\phi)'$. On the one hand, the first conjunct is trivial because $(v, s) \in R(M')^+_{vs,sv}$. On the other hand, we know that for all t, $(t, s) \in R(M')$ iff $(s, t) \in R(M')$. Then by I.H., $\langle W, R(M')^+_{vs,sv}, V \rangle, v \models (\phi)'$ iff $\langle W', R', V', M' \cup \{v\} \rangle, v \models \phi$. Hence, by $\models$ we have $\langle W', R', V', M' \rangle, v \models \textcircled{r}\phi$.

Now suppose the other case, $\langle W, R(M'), V \rangle, v \models \Diamond s$. Then we know $(v, s) \in R(M')$ (by Proposition 5.2.2). By definition of $R(M')$, we have $(s, v) \in R(M')$. Then $v \in M'$, and by I.H. we have $\langle W', R', V', M' \rangle, v \models \textcircled{r}\phi$.

$(\Rightarrow)$ Suppose that φ is satisfiable, i.e., there exists a model $\mathcal{M} = \langle W, R, V, \varnothing \rangle$ and $w \in W$ such that $\langle W, R, V, \varnothing \rangle, w \models \varphi$.

Let s and b be states that do not belong to W. Then we can define the model $\mathcal{M}' = \langle W', R', V' \rangle$ as follows:

$$\begin{array}{ll} W' & = W \cup \{s\} \cup \{b\} \\ R' & = R \cup \{(s,s),(b,b)\} \\ V'(p) & = V(p) \quad \text{for } p \in \mathsf{PROP} \text{ appearing in } \varphi \\ V'(s) & = \{s\} \\ V'(b) & = \{b\}. \end{array}$$

By construction of $\mathcal{M}'$, it is easy to check that $\mathcal{M}', s \models \mathit{Conds}$. Then we can verify that

$$\langle W, R, V, M \rangle, w \models \varphi \ \text{ iff } \ \langle W', R'(M), V' \rangle, s \models \mathsf{Tr}(\varphi),$$

where $R'(M)$ is defined as for the $(\Leftarrow)$ direction of the proof.

Again we need to do structural induction. Boolean cases are easy, and it is also the case for $\textcircled{k}$. If $\langle W, R, V, M \rangle, w \models \Diamond\psi$, then by construction of $\mathcal{M}'$ it is clear that $w \notin V'(s)$ and $w \notin V'(b)$ and $\langle W', R'(M), V' \rangle, v \models (\psi)'$. If $\langle W, R, V, M \rangle, w \models \textcircled{r}\psi$, we can add the edges (w, s) and (s, w) to simulate the storing of w in the memory (if those pairs are in R' means $w \in M$) and continue by evaluating the rest of the translation $(\ )'$ (steps are similar that for the $(\Leftarrow)$ direction of the proof). $\square$

As a consequence of previous theorem, we get:

Theorem 5.2.5. *The satisfiability problem of $\mathcal{ML}(\langle \mathsf{br} \rangle)$ is undecidable.*

Both proofs we presented so far, follow the same ideas to prove undecidability for the satisfiability problem. As we mentioned in the previous section for the $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ case, we conjecture that we can apply the same techniques to prove undecidability for the global version ($\mathcal{ML}(\langle \mathsf{gbr} \rangle)$). We also leave it as future work.

5.3 SWAP LOGIC

Once more, we will simulate the behaviour of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$, i.e., the ability to memorize and check states, without having an external memory. What we have instead is the ability to swap edges in the model. We will build models that contain *switches*, special edges whose position – "off" by default, and "on" if the direction of the edge has been swapped around – will represent whether a point of the model has been remembered with the $\textcircled{r}$ operator. We will simulate the $\textcircled{k}$ predicate by querying the position of these switches. Let us introduce the translation from $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ to $\mathcal{ML}(\langle \mathsf{sw} \rangle)$:

Definition 5.3.1. *Let φ be a formula of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ that does not contain the propositional symbols s and sw. Let $\mathsf{Tr}(\varphi)$ be the following formula:*

$$\begin{array}{lll} (1) & & s \wedge \Box^{(4)} \neg s \\ (2) & \wedge & \neg sw \ \wedge \ \Box \neg sw \ \wedge \ \Box(\ \Diamond(sw \wedge \Box\bot) \ \wedge \ [\mathsf{sw}](sw \to \Box\neg\Diamond sw) \ ) \\ (3) & \wedge & [\mathsf{sw}][\mathsf{sw}]\Box\Box\Box\Box(sw \to \Box\bot) \\ (4) & \wedge & \langle \mathsf{sw} \rangle [\mathsf{sw}](\ \neg s \wedge \neg sw \ \to (\textcircled{r}\Diamond\Diamond(s \wedge \Diamond\textcircled{k}))' \ ) \\ (5) & \wedge & \Diamond(\varphi)'. \end{array}$$

With ( )′ defined as:

$$\begin{array}{lcl}
(ⓡ\psi)' & = & (\ \Diamond sw \ \rightarrow\ \langle \mathsf{sw}\rangle(sw \wedge \Diamond(\psi)')\ )\quad\wedge\quad(\ \neg\Diamond sw\ \rightarrow\ (\psi)'\ ) \\
ⓚ' & = & \neg\Diamond sw \\
(\psi \otimes \chi)' & = & (\psi)' \otimes (\chi)' \quad \textit{for } \otimes \in \{\vee, \wedge\} \\
(\neg\psi)' & = & \neg(\psi)' \\
(\otimes\psi)' & = & \otimes(\neg s \wedge \psi)' \quad \textit{for } \otimes \in \{\Diamond, \langle\mathsf{sw}\rangle\} \\
(\otimes\psi)' & = & \otimes(\neg s \rightarrow \psi)' \quad \textit{for } \otimes \in \{\Box, [\mathsf{sw}]\}.
\end{array}$$

In $\mathsf{Tr}(\varphi)$, the propositional symbol s is used to refer to the evaluation point, that will also be a spy point, i.e., a point that has direct access to all the points in the connected component. sw represents "switch points", they will be used to encode memory operators.

The formula (1) ensures that the propositional symbol s is true at the evaluation point and false at any accessible point between 1 and 4 steps from there. (2) initializes the switches, represented by edges to states where sw is true.

(3) ensures that switch points can be reached from the evaluation point by a unique path. Indeed, if this were not the case, then it would be possible to swap around two edges leading to some switch point, then come back to the evaluation point in two steps by this new path, and come back to the same switch in two steps, where the formula $(sw \wedge \neg\Box\bot)$ would hold.

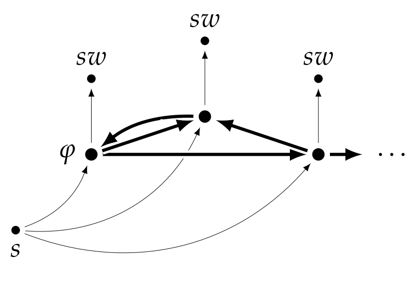
(4) ensures that the evaluation point is linked to every point of the model except itself and the switch points. Note that although $(ⓚ)'$ is true at the evaluation point, the latter is irreflexive by (1), hence $(s \wedge \Diamond ⓚ)'$ ensures an edge goes from the evaluation point to the point where ⓡ occurred.

(5) places the translation of the memory logic formula right after the evaluation point.

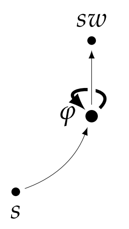
By the definition of $(ⓡ\psi)'$, the action of remembering a point in a model of φ is done in the corresponding model of $\mathsf{Tr}(\varphi)$ by swapping the edge between the corresponding point and its switch point. In the case where the point has already been memorized, i.e., $(ⓚ)' = \neg\Diamond sw$ holds, then nothing needs to be swapped.

It is important that switch points do not have successors and that they have exactly one predecessor. This ensures that the path taken by $(ⓡ\psi)'$ correctly comes back to the same point of the model.

A model of $\mathsf{Tr}(\varphi)$ for some φ is illustrated below:



In this picture, the thick points and lines represent the model of the initial memory logic formula that can be extracted from the whole model. For instance, a model of the formula $\mathsf{Tr}(ⓡ\Diamond ⓚ)$ can be:



Then we can state:

Theorem 5.3.2. *Let φ be a formula of $\mathcal{ML}(ⓡ, ⓚ)$ that does not contain the propositional symbols s and sw. Then, φ and $\mathsf{Tr}(\varphi)$ are equisatisfiable.*

Proof. ($\Rightarrow$) Suppose that φ is satisfiable, i.e., there exists a model $\mathcal{M} = \langle W, R, V, \emptyset\rangle$ and $w \in W$ such that $\langle W, R, V, \emptyset\rangle, w \models \varphi$.

Let sw be a bijective function between W and a set S such that $S \cap W = \emptyset$, and $eval$ a point that is not a member of $S \cup W$. Then we can define the model $\mathcal{M}' = \langle W', R', V'\rangle$ as follows:

$$\begin{array}{ll} W' & = W \cup \{s\} \cup S \\ R' & = R' \cup \{(eval, w) \mid w \in W\} \cup \{(w, sw(w)) \mid w \in W\} \\ V'(p) & = V(p) \quad \text{for } p \in \mathsf{PROP} \text{ appearing in } \varphi \\ V'(s) & = \{s\} \\ V'(sw) & = \{sw(w) \mid w \in W\}. \end{array}$$

Following the description of $\mathsf{Tr}(\varphi)$ given in this section, we can verify that $\mathcal{M}', s \models \mathsf{Tr}(\varphi)$.

($\Leftarrow$) For the other direction, suppose $\mathsf{Tr}(\varphi)$ is satisfiable, i.e., there exists a model $\mathcal{M} = \langle W, R, V\rangle$, and $w \in W$ such that $\langle W, R, V\rangle, w \models \mathsf{Tr}(\varphi)$. Then we can define the model $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$ where

$$\begin{array}{ll} W' & = \{\ v \ \mid\ (w, v) \in R\ \} \\ R' & = R \cap (W' \times W') \\ V'(p) & = V(p) \cap W' \quad \text{for } p \in \mathsf{PROP}. \end{array}$$

We can verify that there exists some $w \in W'$ such that $\mathcal{M}', w \models \varphi$. □

We immediately get:

Theorem 5.3.3. *The satisfiability problem of $\mathcal{ML}(\langle \mathsf{sw}\rangle)$ is undecidable.*

As we did for the other operators, we conjecture that similar ideas can be used to prove the undecidability of the global version of the swap operator.

The technique we use to prove that the satisfiability problem of relation-changing modal logics is undecidable is a reduction of the satisfiability problem of the memory logic $\mathcal{ML}(ⓡ, ⓚ)$. The key is the fact that we can simulate the memorization of an element and its membership to the memory by defining a model with spy points and using relation-changing operators. It looks like the expressive power given by having dynamic behaviour in the language makes it possible to enforce complex structures such as infinite models or models with spy points. This is the reason we conjecture we can use the same techniques using the global operators we study in this thesis, to prove that the logics they give rise are also undecidable (in Chapter 3 we already proved they can enforce infinite models).

6 Model Checking

The biggest difference between time and space is that you can't reuse time.

Merrick Furst.

6.1 The Model Checking Problem

In Chapter 2 we discussed the computational behaviour of modal logics. For $\mathcal{ML}$, the satisfiability problem is PSpace-complete. We showed in previous chapter that the satisfiability problem for the local version of relation-changing modal logics introduced in this thesis is undecidable. We will now analyze the model checking problem: given a model and a formula, check if the model satisfies the formula. This task can be performed in polynomial time for $\mathcal{ML}$. In this section we establish complexity results for the model checking task in the various relation-changing modal logics we presented. All the results are established using a similar argument: hardness proofs are done by encoding the satisfiability problem of Quantified Boolean Formulas (QBF) [Papadimitriou, 1994] as the model checking problem of each logic. While the idea behind the encoding is the same for all the logics involved, the encoding needs to be slightly modified in each case taking into consideration the semantics of the various relation-changing operators. PSpace-hardness for global sabotage was already proved in [Löding and Rohde, 2003b; Löding and Rohde, 2003a], but we provide here a more direct proof. For $\langle \mathsf{sw} \rangle$, the proof was given in [Areces *et al.*, 2013b], and for $\langle \mathsf{sb} \rangle$ and $\langle \mathsf{br} \rangle$, the proof was given in [Areces *et al.*, 2012]

First we will introduce QBF, arguably the most fundamental PSpace-complete problem. Let us introduce its syntax and semantics.

Definition 6.1.1 (QBF Syntax). *Let* PROP *be a countable, infinite set of propositional symbols. Then the set* FORM *of formulas of QBF over* PROP *is defined as:*

$$\mathsf{FORM} ::= x \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x \varphi,$$

where $x \in$ PROP and $\varphi, \psi \in$ FORM.

Given that satisfiability depends only on assignments of truth to propositional variables, QBF models are valuations (functions that assign truth values to the variables).

Definition 6.1.2 (QBF Semantics). *Let $v : \{x_1, \ldots, x_k\} \to \{0,1\}$ be a valuation, the relation $\models_{\text{qbf}}$ is defined as:*

$$\begin{array}{lll} v \models_{\text{qbf}} x_i & \textit{iff} & v(x_i) = 1 \\ v \models_{\text{qbf}} \neg\varphi & \textit{iff} & v \not\models_{\text{qbf}} \varphi \\ v \models_{\text{qbf}} \varphi \wedge \psi & \textit{iff} & v \models_{\text{qbf}} \varphi \textit{ and } v \models_{\text{qbf}} \psi \\ v \models_{\text{qbf}} \exists x_i \varphi & \textit{iff} & v[x_i \mapsto 0] \models_{\text{qbf}} \varphi \textit{ or } v[x_i \mapsto 1] \models_{\text{qbf}} \varphi \end{array}$$

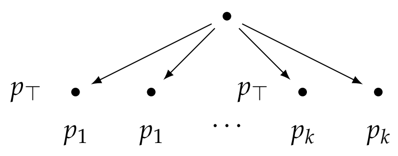
The valuation $v[x_i \mapsto n]$ is defined as $v[x_i \mapsto n](x_i) = n$ and $v[x_i \mapsto n](x) = v(x)$, for $x \neq x_i$. φ is satisfiable if there is a valuation v such that $v \models_{\text{qbf}} \varphi$.

The satisfiability problem for QBF is PSpace-complete [Papadimitriou, 1994]. We will use this result to prove that model checking problem for all the relation-changing logics we introduced in this thesis is PSpace-hard.

Theorem 6.1.3. *For $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$, model checking for any of the logics $\mathcal{ML}(\blacklozenge)$ is PSpace-hard.*

Proof. We will reduce the PSpace-complete satisfiability problem of QBF to the model checking problem of each of these logics. We will give a complete proof for the $\langle \mathsf{sw} \rangle$ case; for the other operators a similar strategy establishes the result.

- Consider $\mathcal{ML}(\langle \mathsf{sw} \rangle)$. Let α be a QBF formula with variables $\{x_1, \ldots, x_k\}$. Without loss of generality we can assume that α has no free variables and no variable is quantified twice. One can build in polynomial time the relational structure $\mathcal{M}_k$ showed below. The evaluation point has two successors for each variable in α, but just one of each kind is labeled by $p_\top$.



  $\mathcal{M}_k = \langle W, R, V \rangle$ is built over a signature with one relational symbol and propositions $\{p_\top, p_1, \ldots, p_k\}$, where:

$$\begin{array}{ll} W & = \{w\} \cup \{w_i^1, w_i^0 \mid 1 \leq i \leq k\} \\ V(p_i) & = \{w_i^1, w_i^0\} \\ V(p_\top) & = \{w_i^1 \mid 1 \leq i \leq k\} \\ R & = \{(w, w_i^1), (w, w_i^0) \mid 1 \leq i \leq k\}. \end{array}$$

  Now we need to translate quantified boolean formulas to $\mathcal{ML}(\langle \mathsf{sw} \rangle)$-formulas. The idea is that each time we have an existential formula over a variable x_i, we swap around one of the edges pointing to p_i states in $\mathcal{M}_k$. We continue the evaluation of the rest of the formula at this p_i point in the new model variant. Swapping the $p_\top$ one, represents that the truth value has to be assigned to 1. Otherwise, x_i is assigned to 0. Other QBF operators are translated in the obvious way.

Let $(\ )'$ be the following linear translation from QBF to $\mathcal{ML}(\langle \mathsf{sw} \rangle)$:

$$\begin{array}{ll} (\exists x_i.\alpha)' &= \langle \mathsf{sw} \rangle (p_i \wedge \Diamond(\alpha)') \\ (x_i)' &= \neg\Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' &= \neg(\alpha)' \\ (\alpha \wedge \beta)' &= (\alpha)' \wedge (\beta)'. \end{array}$$

It remains to be proved that α is satisfiable iff $\mathcal{M}_k, w \models (\alpha)'$ holds. For a model $\mathcal{M}$ with relation R we define $v_R : \{x_1, \ldots, x_k\}$ as "$v_R(x_i) = 1$ iff $(w, w_i^1) \notin R$", in the present case, iff the link between w and w_i^1 has been swapped.

Let β be any subformula of α. We will show by induction on β that $\mathcal{M}, w \models (\beta)'$ iff $v_R \models_{\mathrm{qbf}} \beta$. The first observation is that R satisfies i) if x_i is free in β, then $(w, w_i^1) \notin R$ or $(w, w_i^0) \notin R$ but not both, and ii) if x_i is not free in β then $(w, w_i^1) \in R$ and $(w, w_i^0) \in R$. From here it will follow that $\mathcal{M}_k, w \models (\alpha)'$ iff $v \models_{\mathrm{qbf}} \alpha$ for any v since α has no free variables, iff α is satisfiable.

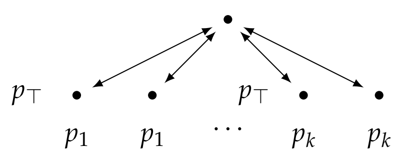
For the base case, $v_R \models_{\mathrm{qbf}} x_i$ iff $(w, w_i^1) \notin R$ which implies (from the definition of $\mathcal{M}_k$) $\mathcal{M}, w \models (x_i)'$. For the other direction, suppose $\mathcal{M}, w \not\models (x_i)'$. Hence $\mathcal{M}, w \models \Diamond(p_i \wedge p_\top)$ which implies $(w, w_i^1) \in R$ and $v_R \not\models_{\mathrm{qbf}} x_i$.

The Boolean cases follow directly from the inductive hypothesis.

Consider the case $\beta = \exists x_i.\gamma$. Since no variable is bound twice in α we know $(w, w_i^1) \in R$ and $(w, w_i^0) \in R$. We have $v_R \models_{\mathrm{qbf}} \beta$ iff $(v_R[x_i \mapsto 0] \models_{\mathrm{qbf}} \gamma$ or $v_R[x_i \mapsto 1] \models_{\mathrm{qbf}} \gamma)$ iff $(v_{R^*_{w_i^0 w}} \models_{\mathrm{qbf}} \gamma$ or $v_{R^*_{w_i^1 w}} \models_{\mathrm{qbf}} \gamma)$. By inductive hypothesis, this is the case if and only if $(\mathcal{M}^*_{w_i^0 w}, w_i^0 \models \Diamond(\gamma)'$ or $\mathcal{M}^*_{w_i^1 w}, w_i^1 \models \Diamond(\gamma)')$ iff $\mathcal{M}, w \models \langle \mathsf{sw} \rangle (p_i \wedge \Diamond(\gamma)')$ iff $\mathcal{M}, w \models (\exists x_i.\gamma)'$.

This shows that the model checking problem of $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ is PSpace-hard.

- For $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ and $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, we use the following model $\mathcal{M}_k$:



$\mathcal{M}_k = \langle W, R, V \rangle$ is built as for the swap case, but with:

$$\begin{array}{ll} W &= \{w\} \cup \{w_i^1, w_i^0 \mid 1 \leq i \leq k\} \\ V(p_i) &= \{w_i^1, w_i^0\} \\ V(p_\top) &= \{w_i^1 \mid 1 \leq i \leq k\} \\ R &= \{(w, w_i^1), (w, w_i^0), \\ & \quad\ (w_i^1, w), (w_i^0, w) \mid 1 \leq i \leq k\} \end{array}$$

Now we will give the translations from QBF to local and global sabotage logics. The lack of an edge pointing from the evaluation point to an $p_\top$ point means that the corresponding variable has to be assigned to 1, otherwise to 0.

Let $(\ )'$ be the following linear translation from QBF to $\mathcal{ML}(\langle \mathsf{sb} \rangle)$:

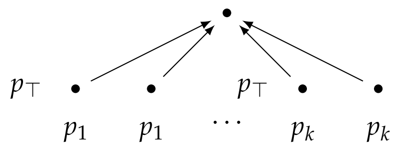
$$\begin{array}{ll}(\exists x_i.\alpha)' &= \langle \mathsf{sb} \rangle (p_i \wedge \Diamond(\alpha)') \\ (x_i)' &= \neg\Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' &= \neg(\alpha)' \\ (\alpha \wedge \beta)' &= (\alpha)' \wedge (\beta)'.\end{array}$$

From QBF to $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, we provide the following translation:

$$\begin{array}{ll}(\exists x_i.\alpha)' &= \langle \mathsf{gsb} \rangle ((\neg\Diamond(p_i \wedge p_\top) \vee \neg\Diamond(p_i \wedge \neg p_\top)) \wedge \Diamond(p_i \wedge \Diamond(\alpha)')) \\ (x_i)' &= \neg\Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' &= \neg(\alpha)' \\ (\alpha \wedge \beta)' &= (\alpha)' \wedge (\beta)'.\end{array}$$

In both cases, showing that a QBF formula α is satisfiable if, and only if, $\mathcal{M}_k, w \models (\alpha)'$ holds can be done similarly to the case of $\mathcal{ML}(\langle \mathsf{sw} \rangle)$.

- To prove PSpace-hardness for $\mathcal{ML}(\langle \mathsf{br} \rangle)$ and $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$, build the following $\mathcal{M}_k$:



The model $\mathcal{M}_k = \langle W, R, V \rangle$ is defined as follows:

$$\begin{array}{ll}W &= \{w\} \cup \{w_i^1, w_i^0 \mid 1 \leq i \leq k\} \\ V(p_i) &= \{w_i^1, w_i^0\} \\ V(p_\top) &= \{w_i^1 \mid 1 \leq i \leq k\} \\ R &= \{(w_i^1, w), (w_i^0, w) \mid 1 \leq i \leq k\}\end{array}$$

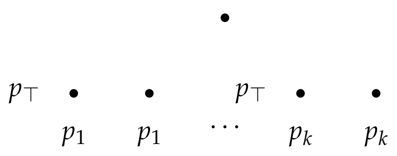
For $\mathcal{ML}(\langle \mathsf{br} \rangle)$ we use the following linear translation $(\ )'$, which puts a new edge to the corresponding $p_\top$ point to assign the variable to 1, or to the $\neg p_\top$ point to assign 0:

$$\begin{array}{ll}(\exists x_i.\alpha)' &= \langle \mathsf{br} \rangle (p_i \wedge \Diamond(\alpha)') \\ (x_i)' &= \Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' &= \neg(\alpha)' \\ (\alpha \wedge \beta)' &= (\alpha)' \wedge (\beta)'.\end{array}$$

For $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ we use the following linear translation $(\ )'$, following the same idea as for $\mathcal{ML}(\langle \mathsf{br} \rangle)$ but swapping the corresponding edge and staying in the same evaluation point (is a global operator):

$$\begin{array}{ll}(\exists x_i.\alpha)' &= \langle gsw \rangle (\Diamond p_i \ \wedge \ (\alpha)') \\ (x_i)' &= \Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' &= \neg(\alpha)' \\ (\alpha \wedge \beta)' &= (\alpha)' \wedge (\beta)'.\end{array}$$

- Finally, to prove PSpace-hardness for $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$, build the following model $\mathcal{M}_k$:



$\mathcal{M}_k = \langle W, R, V \rangle$ is defined as:

$$\begin{array}{ll} W & = \{w\} \cup \{w_i^1, w_i^0 \mid 1 \leq i \leq k\} \\ V(p_i) & = \{w_i^1, w_i^0\} \\ V(p_\top) & = \{w_i^1 \mid 1 \leq i \leq k\} \\ R & = \emptyset \end{array}$$

And consider the following linear translation $(\ )'$ from QBF to $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$:

$$\begin{array}{ll} (\exists x_i.\alpha)' & = \langle gbr \rangle (\Diamond p_i \ \wedge \ (\alpha)') \\ (x_i)' & = \Diamond (p_i \wedge p_\top) \\ (\neg \alpha)' & = \neg (\alpha)' \\ (\alpha \wedge \beta)' & = (\alpha)' \wedge (\beta)'. \end{array}$$

We have reduced the QBF satisfiability problem to the model checking problem for each of the relation-changing modal logics of Definition 3.2.1. Then, model checking for all of them is PSpace-hard. □

Now, we will prove PSpace-completeness for the model checking problem of the six logics that we are investigating. It remains to show that checking that a formula holds in a model can be done using polynomial space.

Theorem 6.1.4. *Model checking for* $\mathcal{ML}(\langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle)$ *is in* PSpace.

Proof. The evaluation of the truth of a formula in a model can be done by a polynomial space algorithm that follows Definition 3.2.3.

The algorithm works on the same copy of the model, except when dealing with formulas whose main connector is $\langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{br} \rangle$ or $\langle \mathsf{gbr} \rangle$ (i.e., relation-changing operators). In such cases, by proceeding depth-first among at most $|W|$ possible choices (i.e., the size of the set of states), the algorithm only allocates as much additional space as the size of the initial model to store the modified copy. This memory can be reclaimed once the result of the recursive call is known. The maximum number of copies of the input model in memory is bounded by the nesting of relation-changing operators of the input formula. Hence the algorithm runs using only polynomial space. □

With the previous results we get:

Theorem 6.1.5. *For* $\blacklozenge \in \{\langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$, *model checking for any of the logics* $\mathcal{ML}(\blacklozenge)$ *is* PSpace-*complete.*

The last theorem establishes that model checking for relation-changing modal logics is as hard as model checking for $\mathcal{FOL}$. The challenge now, is to find lower complexities for some sub-problems, as it has be done for $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$ in [Löding and Rohde, 2003a; Rohde, 2006]. Besides the high expressivity of the languages introduced in this thesis, it is still possible to find some tractable reasoning tasks, such as model checking against a fixed model or against a fixed formula. We will study these problems in the next section.

6.2 FORMULA COMPLEXITY AND PROGRAM COMPLEXITY

We established the complexity of the *combined* model checking task, measured as a function of the length of an input model and an input formula. It is also possible to consider the task of model checking against a fixed model, measuring its complexity as a function of the size of an input formula (this is known as the *formula* complexity). One can also fix a formula and measure the complexity of model checking as a function of the length of an input model (known as the *program complexity* or *data complexity*). For many logics, solving these two problems is as hard as solving the combined model checking problem. This is the case of $\mathcal{FOL}$: it was proved in [Vardi, 1982] that program and formula complexity for $\mathcal{FOL}$ is PSpace-complete. On the other hand, while combined model checking and formula complexity for LTL and CTL* are PSpace-complete, the program complexity of both logics is NLogSpace-complete (see [Vardi and Wolper, 1986; Kupferman *et al.*, 2000; Schnoebelen, 2002] for details).

It has been shown in [Löding and Rohde, 2003a; Rohde, 2006] that the formula complexity and the program complexity of $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$ are respectively linear and polynomial. We are going to show that these results generalize to $\mathcal{ML}(\langle\mathsf{sb}\rangle)$, $\mathcal{ML}(\langle\mathsf{sw}\rangle)$, $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$, $\mathcal{ML}(\langle\mathsf{br}\rangle)$ and $\mathcal{ML}(\langle\mathsf{gbr}\rangle)$. In fact, we extend these results to many more dynamic operators, granted they behave in a controlled way. As we want to give a general result that generalizes the particular case of the six logics introduced in this thesis, we need also to generalize the definition of model variant. We can introduce a family of model update functions and define the semantics of the operators based on them. Hence, we can instantiate the results that we will prove in this section with every operator than can be defined with model update functions.

Let W be the set of states of a model and let f be a function that takes an element w of W and the current accessibility relation R over W and returns a set of pairs (v, S), where $v \in W$ is the new point of evaluation and S is the new accessibility relation to be used (here we will exclusively discuss binary accessibility relations and, hence, $f : W \times 2^{W^2} \mapsto 2^{W \times 2^{W^2}}$ but, of course, the idea generalizes to modalities of arbitrary arity). Each different function f defines a relation-changing operator. For example, $\langle\mathsf{sw}\rangle$ would be defined by a function f such that $f(w, R) = \{(v, R \backslash \{(w, v)\} \cup \{(v, w)\}) \mid (w, v) \in R\}$.

Clearly, the modalities defined in this form do not cover all possible dynamic modal operators (e.g., dynamic modal operators investigated in different dynamic epistemic logics [van Ditmarsch *et al.*, 2007] change the set of states in the model, or the valuation function; see [Areces and Gorín, 2010] for an even more general, but complex framework). But, as we saw, this framework covers van Benthem's original sabotage operator and other variants investigated in, for instance, [Rohde, 2006; Areces *et al.*, 2012; Areces *et al.*, 2013b; Areces *et al.*, 2013c].

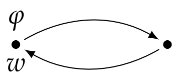
Definition 6.2.1 (Model update functions)**.** *Let $\mathcal{C}$ be a class of models. We say that* $F = \{f_W : W \times 2^{W^2} \mapsto 2^{W \times 2^{W^2}} \mid \mathcal{M} = \langle W, R, V \rangle \in \mathcal{C}\}$ *is a family of* model update functions. *We say that $\mathcal{C}$ is closed under a family of model update functions F if whenever $\mathcal{M} = \langle W, R, V \rangle \in \mathcal{C}$, then $\{\langle W, R', V \rangle \mid f_W \in F, w \in W, (v, R') \in f_W(w, R)\} \subseteq \mathcal{C}$.*

Each dynamic operator is defined on a particular domain W by a certain model update function f_W. Clearly, the class of all pointed models is closed under any family of model update functions. In the rest of the thesis we will only discuss the class of all models, and we will pay particular attention to a number of families of model update functions which can be simply defined. We are talking about the model variants defined in Chapter 3.

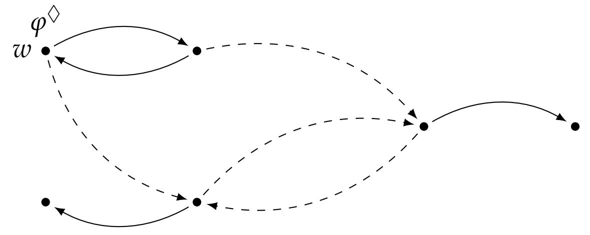
We can now use these basic updating operations over relations and models to define model update functions, and their respective families. Each of these families will give rise to a natural dynamic modal operator. For instance, we can define the six relation-changing modal operators investigated in this thesis:

$$
\begin{array}{lll}
F_{sw} = \{f_W^{sw}\}, \text{ where for any } \mathcal{M} = \langle W, R, V \rangle, f_W^{sw}(w, R) & = & \{(v, R^*_{vw}) \mid wv \in R\}. \\
F_{gsw} = \{f_W^{gsw}\}, \text{ where for any } \mathcal{M} = \langle W, R, V \rangle, f_W^{gsw}(w, R) & = & \{(w, R^*_{vu}) \mid uv \in R\}. \\
F_{sb} = \{f_W^{sb}\}, \text{ where for any } \mathcal{M} = \langle W, R, V \rangle, f_W^{sb}(w, R) & = & \{(v, R^-_{wv}) \mid wv \in R\}. \\
F_{gsb} = \{f_W^{gsb}\}, \text{ where for any } \mathcal{M} = \langle W, R, V \rangle, f_W^{gsb}(w, R) & = & \{(w, R^-_{uv}) \mid uv \in R\}. \\
F_{br} = \{f_W^{br}\}, \text{ where for any } \mathcal{M} = \langle W, R, V \rangle, f_W^{br}(w, R) & = & \{(v, R^+_{wv}) \mid wv \notin R\}. \\
F_{gbr} = \{f_W^{gbr}\}, \text{ where for any } \mathcal{M} = \langle W, R, V \rangle, f_W^{gbr}(w, R) & = & \{(w, R^+_{uv}) \mid uv \notin R\}.
\end{array}
$$

To investigate formula complexity and program complexity, we will use the technique of model unfolding. Say we start with a model $\mathcal{M}$ and a formula φ of a logic $\mathcal{ML}(\blacklozenge)$:



Starting from $\mathcal{M}$, we explicitly build the variants obtained by successive applications of the relation-changing operator $\blacklozenge$.



In this unfolded model, we use two distinct relations: one to represent the relations as they should be in each model variant, and another as the external relation that links model variants between them. At the syntactic level, we rewrite φ into a basic modal formula $\varphi^{\Diamond}$ with two classic modalities. Following this idea we reduce the model checking problem of a dynamic language to the model checking problem of the basic modal logic, which we have showed in Theorem 1.2.10 is polynomial with respect to the size of the formula and the model. It can be solved in time $\mathcal{O}(|\varphi| \cdot |\mathcal{M}|)$, where $|\varphi|$ is the size of the given $\mathcal{ML}$-formula φ and $|\mathcal{M}|$ is the size of the given model $\mathcal{M}$.

We define the following translation from relation-changing modal logics to modal logic with two modalities:

Definition 6.2.2. *Consider the language $\mathcal{ML}(\blacklozenge_F)$ for some family of model update functions F. Define the following translation from this language to the basic modal language with two modalities:*

$$\begin{array}{rcl} (p)^{\Diamond} & = & p \\ (\varphi \wedge \psi)^{\Diamond} & = & \varphi^{\Diamond} \wedge \psi^{\Diamond} \\ (\neg\varphi)^{\Diamond} & = & \neg\varphi^{\Diamond} \\ (\Diamond\varphi)^{\Diamond} & = & \Diamond_1\varphi^{\Diamond} \\ (\blacklozenge_F\varphi)^{\Diamond} & = & \Diamond_2\varphi^{\Diamond}. \end{array}$$

The previous translation represents the effect of the relation-changing operators as a new modality which moves the evaluation point to a different model variant. We can then apply the model checking algorithm for $\mathcal{ML}$ to this kind of formulas. Before we complete the unfolding by constructing the corresponding unfolded model, we need to introduce a measure of the complexity of relation-changing modal logics.

Definition 6.2.3. *The maximum number of nested dynamic modalities is called the* dynamic depth, *and is defined as:*

$$\begin{array}{rcl} dd(p) & = & 0 \\ dd(\neg\varphi) & = & dd(\varphi) \\ dd(\varphi \wedge \varphi') & = & max(dd(\varphi), dd(\varphi')) \\ dd(\Diamond\varphi) & = & dd(\varphi) \\ dd(\blacklozenge_F\varphi) & = & 1 + dd(\varphi), \end{array}$$

with $\blacklozenge_F$ the relation-changing operator defined by the family of model update functions F.

Measuring formulas according to the number of nested relation-changing modalities will help us establish some bounds in the size of the unfolded models. The general idea is that the dynamic depth will be the number of model variants that we will put in the unfolded model. These models will be constructed using the following definition of all possible accessibility relations from a model update function:

Definition 6.2.4. *Let W be a domain, $R \subseteq W^2$ an accessibility relation, f a model update function for W, and n a natural number. We define inductively $Vars(R, f, n)$, the set of all possible relation variants obtained applying n times the function f on the initial relation R as:*

$$\begin{array}{rcl} Vars(R, f, 0) & = & \{R\} \\ Vars(R, f, n+1) & = & \{T \mid (v, T) \in f(w, S), S \in Vars(R, f, n), w \in W\}. \end{array}$$

Let

$$Vars(R, f) = \bigcup_{n<\omega} Vars(R, f, n)$$

be the set of all relation variants obtained applying f on the initial relation R.

We now look into the details of model unfolding. We will introduce the translation from relation-changing models to static models based on model update functions. These static models will be used to model check unfolded formulas of Definition 6.2.2.

Definition 6.2.5. *Let $\mathcal{M} = \langle W, R, V\rangle$. Let $f : W \times 2^{W^2} \mapsto 2^{(W\times 2^{W^2})}$.*
We define $\mathcal{M}_{f,0} = \langle W', \{R'_1, R'_2\}, V'\rangle$ as:

$$\begin{array}{lcl} W' & = & W \times \{R\} \\ R'_1 & = & \{((s,R),(t,R)) \mid (s,t) \in R\} \\ R'_2 & = & \emptyset \\ V'(p) & = & \{(s,S) \mid s \in V(p), S \in Vars(R,f)\}. \end{array}$$

We define $\mathcal{M}_{f,n} = \langle W', \{R'_1, R'_2\}, V'\rangle$ (for $n \geq 1$) as:

$$\begin{array}{lcl} W' & = & W \times Vars(R,f,n) \\ R'_1 & = & \bigcup\{((s,S),(t,S)) \mid (s,t) \in S, S \in Vars(R,f)\} \\ R'_2 & = & \{((s,S),(t,T)) \mid (t,T) \in f(s,S), S \in Vars(R,f,n-1)\} \\ V'(p) & = & \{(s,S) \mid s \in V(p), S \in Vars(R,f)\}. \end{array}$$

Finally, we define $\mathcal{M}_f = \langle W', \{R'_1, R'_2\}, V'\rangle$ as:

$$\begin{array}{lcl} W' & = & W \times Vars(R,f) \\ R'_1 & = & \{((s,S),(t,S)) \mid (s,t) \in S, S \in Vars(R,f)\} \\ R'_2 & = & \{((s,S),(t,T)) \mid (t,T) \in f(s,S), S \in Vars(R,f)\} \\ V'(p) & = & \{(s,S) \mid s \in V(p), S \in Vars(R,f)\}. \end{array}$$

The idea is that a relation-changing model satisfies a formula if and only if after doing the unfolding of the model and the formula, satisfiability is preserved. It is easy to check the following, according to the previous definitions:

Lemma 6.2.6. *Let $\mathcal{M} = \langle W, R, V\rangle$, and φ a formula. Then*

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}_{f,dd(\varphi)}, (w,R) \models \varphi^{\Diamond}.$$

Now for the complexity result. We first observe that if a relation-changing operator can create only a polynomial number of relation variants in one step, then the number of possible relation variants after n steps is also polynomial:

Lemma 6.2.7. *Let $f : W \times 2^{W^2} \mapsto 2^{(W\times 2^{W^2})}$. Suppose that for all $s \in W$, $S \subseteq W\times W$, there exists $c > 0$ such that $|f(s,S)| \in \mathcal{O}(|W|^c)$. Then there exists $d > 0$ such that $|Vars(R,f,n)| \in \mathcal{O}(|W|^{dn})$.*

In particular, given a model $\mathcal{M}$, it is possible to build $\mathcal{M}_{f,n}$ in polynomial time in terms of n, where n is the number of applications of f in $\mathcal{M}$.

This leads to the general result of formula complexity and program complexity for modal logics equipped with "polynomial" relation-changing operators:

Theorem 6.2.8. *Let $\mathcal{L}$ be the basic modal logic equipped with a relation-changing operator whose semantics is defined by a family F of model update functions such that for all $\mathcal{M} = \langle W, R, V\rangle$, all $s \in W$, and all $S \subseteq W \times W$, $|f(s,S)| \in \mathcal{O}(|W|^c)$.*

1. *The model checking problem for $\mathcal{L}$ with a fixed finite model can be solved in linear time with respect to the size of the formula (formula complexity).*

2. *The model checking problem for $\mathcal{L}$ with a fixed formula can be solved in polynomial time with respect to the size of the finite model (program complexity).*

Proof. Both parts rely on the complexity of model checking for $\mathcal{ML}$ (Theorem 1.2.10).

1. Fix a model $\mathcal{M} = \langle W, R, V \rangle$ with a state $w \in W$. For some input formula φ, build $\varphi^{\Diamond}$ in linear time with respect to $|\varphi|$. Then check that $\mathcal{M}_f, (w, R) \models \varphi^{\Diamond}$ in time linear with respect to $|\varphi^{\Diamond}| = |\varphi|$.

2. Fix a formula φ, with $dd(\varphi) = n$. For some input model $\mathcal{M} = \langle W, R, V \rangle$ and state $w \in W$, build $\mathcal{M}_{f,n}$ in polynomial time with respect to $|\mathcal{M}|$ (Lemma 6.2.7). Then check that $\mathcal{M}_{f,n}(w, R) \models \varphi^{\Diamond}$ in linear time with respect to $\left|\mathcal{M}_{f,n}\right|$, i.e., in polynomial time with respect to $|\mathcal{M}|$.

□

Observe that the modalities $\langle \mathsf{sw} \rangle$, $\langle \mathsf{gsw} \rangle$, $\langle \mathsf{sb} \rangle$, $\langle \mathsf{gsb} \rangle$, $\langle \mathsf{br} \rangle$ and $\langle \mathsf{gbr} \rangle$ all satisfy the condition of Lemma 6.2.7, hence Theorem 6.2.8 applies to the corresponding logics. Moreover, the result extends to the basic modal logic equipped with any combination of relation-changing operators if each of them satisfies the conditions of the lemma.

A simple example of a relation-changing modal operator that does not satisfy the conditions of the theorem would be the following "universal-universal modality", which can blow up the number of possible relations to 2^{W^2} in just one step:

$$f_W : (s, S) \mapsto \{(t, T) \mid T \subseteq W \times W, t \in W\}.$$

Evaluating this "universal-universal modality" consists in considering all models with domain W, some fixed valuation V and all possible binary relations on W.

7 Tableaux

In solving a problem of this sort, the grand thing is to be able to reason backwards. That is a very useful accomplishment, and a very easy one, but people do not practise it much. In the every-day affairs of life it is more useful to reason forwards, and so the other comes to be neglected. There are fifty who can reason synthetically for one who can reason analytically... Let me see if I can make it clearer. Most people, if you describe a train of events to them, will tell you what the result would be. They can put those events together in their minds, and argue from them that something will come to pass. There are few people, however, who, if you told them a result, would be able to evolve from their own inner consciousness what the steps were which led up to that result. This power is what I mean when I talk of reasoning backwards, or analytically.

-Sherlock Holmes.

from "A Study in Scarlet" , Sir Arthur Conan Doyle.

7.1 TABLEAU CALCULUS

Besides theoretical results with respect to reasoning tasks, we investigate concrete procedures to implement them. Even though we have already proved that the satisfiability problem for many of the relation-changing modal logics we introduced is undecidable, we can define non-terminating procedures for the satisfiability of a formula. Several decision procedures have been investigated for modal logics, but *tableau algorithms* are the best known and most used in implementations. Tableaux for first-order logic were first introduced in [Beth, 1955] and investigated later in [Smullyan, 1968]. First results on tableaux for modal logics were introduced in [Fitting, 1972].

A tableau algorithm is a procedure that decides satisfiability of a formula by exploring the satisfiability of its subformulas. The procedure takes a formula as input, and decides if it is satisfiable or not. In the case the answer is "yes", the algorithm also returns a model for the formula. For this reason tableaux is not just a decision procedure, it can also be considered as a model building method. For further details see [D'Agostino *et al.*, 1999].

We present basic definitions for different tableau algorithms for the relation-changing modal logics we introduced in this thesis, based on the results presented in [Areces *et al.*, 2013c]. These algorithms will rely on the same data structures and will only differ in some of their rules.

Definition 7.1.1 (Tableau formulas)**.** *Let* NOM *be an infinite, well ordered set of symbols we call* nominals. *A* tableau formula *is either a* prefixed formula*, an* equational formula *or a* relational formula. *A prefixed formula is of the form* $(n, X) : \varphi$*, with* $n \in \mathsf{NOM}$,

$X \subseteq \mathsf{NOM}^2$, and φ a formula of the considered object language. An equational formula is a Boolean combination of formulas of the form $n \dot{=} m$ or $n \dot{\neq} m$ for $n, m \in \mathsf{NOM}$. We also use the following notation:

$$\begin{array}{lcllcl} & & & nm \dot{\in} X & := & \bigvee_{xy \in X} nm \dot{=} xy \\ nm \dot{=} xy & := & n \dot{=} x \wedge m \dot{=} y & & & \\ nm \dot{\neq} xy & := & n \dot{\neq} x \vee m \dot{\neq} y & nm \dot{\notin} X & := & \bigwedge_{xy \in X} nm \dot{\neq} xy. \end{array}$$

In particular $nm \dot{\in} \emptyset$ is a notation for $\perp$ and $nm \dot{\notin} \emptyset$ is a notation for $\top$. A relational formula is of the form $\dot{R}nm$ or $\neg \dot{R}nm$, with $n, m \in \mathsf{NOM}$.

The set X of a prefixed formula $(n, X) : \varphi$ is used to describe the model variant in which the formula φ is to be interpreted. According to the logic we are in, this set is to be interpreted differently. This is done by fixing a function f that, out of relations $R, S \subseteq W \times W$ yields another relation $R' = f(R, S)$.

Definition 7.1.2 (Branches and interpretations). *A* branch *is a non-empty set of tableau formulas. Let $\mathcal{M} = \langle W, R, V \rangle$ be a model, $f : W^2 \times W^2 \to W^2$ a relation-changing function and $\sigma : \mathsf{NOM} \to W$ a mapping from nominals to states of $\mathcal{M}$. Let $X^\sigma = \{\sigma(a)\sigma(b) \mid ab \in X\}$, for $X \subseteq \mathsf{NOM}^2$.*

Given $\mathcal{M} = \langle W, R, V \rangle$, let $\mathcal{M}^f_{X^\sigma} = \langle W, f(R, X^\sigma), V \rangle$. That is, $\mathcal{M}^f_{X^\sigma}$ is the model $\mathcal{M}$ updated by the relation-changing function f according to a set of pairs of nominals X under mapping σ.

A branch Θ is satisfiable *if there exists a model $\mathcal{M} = \langle W, R, V \rangle$ and a mapping σ such that all the formulas of Θ are satisfiable under model $\mathcal{M}$ and mapping σ. That is, they should satisfy the following conditions:*

- *if $(n, X) : \varphi \in \Theta$ then $\mathcal{M}^f_{X^\sigma}, \sigma(n) \models \varphi$,*
- *if $n \dot{=} m \in \Theta$ then $\sigma(n) = \sigma(m)$,*
- *if $n \dot{\neq} m \in \Theta$ then $\sigma(n) \neq \sigma(m)$,*
- *Boolean combinations of equational formulas are interpreted as expected,*
- *if $\dot{R}nm \in \Theta$ then $R\sigma(n)\sigma(m)$,*
- *if $\neg \dot{R}nm \in \Theta$ then $\neg R\sigma(n)\sigma(m)$.*

A branch is unsatisfiable *if it is not satisfiable.*

Now we have introduced the data structures that will be used, we can define the calculus formally.

Definition 7.1.3 (Tableau). *A* tableau calculus *is a set of rules such that each rule applies to a branch and yields one or more branches, under certain conditions. These conditions are called saturation conditions, and stipulate that no rule can be applied twice on the same premises, and that no formula can be introduced twice in a branch.*

A tableau *is a tree in which each node defines a tableau branch, and edges represent applications of tableau rules. A tableau is expanded as much as possible by the rules of the* *system (i.e., rules are applied whenever possible according to the saturation condition). A fully expanded branch is called* saturated*.*

A tableau branch is closed *if it contains $\bot$, otherwise it is* open*. A tableau is closed if all branches are closed, otherwise it is open.*

Given a branch Θ, $\sim_\Theta$ denotes the equivalence closure of the relation $\{nm \mid n \doteq m \in \Theta\}$, and we write $\bar{n}$ for the smallest nominal x such that $x \sim_\Theta n$. For $X \subseteq \mathsf{NOM}^2$ we write $\bar{X} = \{\bar{n}\bar{m} \mid nm \in X\}$. Figure 11 presents the rules common to all the tableau calculi presented in this chapter. They are the Boolean rules $(\wedge)$ and $(\vee)$, the clashing rules $(\bot_{atom})$ and $(\bot_{\neq})$, the equational rules $(R{\sim})$ and (Id), and the unrestricted blocking rule (ub) [Schmidt and Tishkovsky, 2007]. We use the unrestricted blocking rule as a way to saturate branches with equational formulas. These formulas can appear as premises of tableau rules in the calculi we introduce later.

$$\frac{(n,X):\ \varphi\wedge\psi}{\begin{array}{c}(n,X):\varphi\\ (n,X):\psi\end{array}}\ (\wedge) \qquad\qquad \frac{(n,X):\ \varphi\vee\psi}{(n,X):\varphi \,\big|\, (n,X):\psi}\ (\vee)$$

$$\frac{\begin{array}{c}(n,X_1):p\\ (n,X_2):\neg p\end{array}}{\bot}\ (\bot_{atom})^1 \qquad\qquad \frac{\begin{array}{c}n\sim_\Theta m\\ n\dot{\neq} m\end{array}}{\bot}\ (\bot_{\neq})$$

$$\frac{\dot{R}nm}{\dot{R}\bar{n}\bar{m}}\ (R{\sim}) \qquad \frac{(n,X):\varphi}{(\bar{n},X):\varphi}\ (Id) \qquad \frac{}{n\doteq m \ \mid\ n\dot{\neq} m}\ (ub)^2$$

^{1} $p \in \mathsf{PROP}$.
^{2} n and m are two different nominals in the branch.

Figure 11: Common tableau rules.

This result follows easily from the tableau rules:

Lemma 7.1.4. *Let Θ be a saturated open branch. If $nm\dot{\in}S$ is in Θ then $\bar{n}\bar{m} \in \bar{S}$. If $nm\dot{\notin}S$ is in Θ then $\bar{n}\bar{m} \notin \bar{S}$.*

When it comes to adequacy of a tableau calculus, we have to consider two properties: completeness and soundness. Given a tableau calculus $\mathcal{T}$, let us write $\mathcal{T}(\varphi)$ to refer to a tableau obtained by running $\mathcal{T}$ on the input formula $(n_0, \emptyset) : \varphi$, where n_0 is the smallest nominal in NOM. Then we define:

Definition 7.1.5 (Completeness). *A tableau calculus $\mathcal{T}$ is* complete *if for any formula φ, if $\mathcal{T}(\varphi)$ is open then φ is satisfiable.*

Definition 7.1.6 (Soundness). *A tableau calculus $\mathcal{T}$ is* sound *if for any formula φ, if φ is satisfiable then $\mathcal{T}(\varphi)$ is open.*

We define models induced from open branches.

Definition 7.1.7 (Induced Models). *Let Θ be an open branch. We define the* induced model for Θ *as* $\mathcal{M}^{\Theta} = \langle W^{\Theta}, R^{\Theta}, V^{\Theta}\rangle$*, where:*

$$\begin{array}{lcl} W^{\Theta} & = & \{\bar{n} \mid n \in \Theta\} \\ R^{\Theta} & = & \{(\bar{n}, \bar{m}) \mid \dot{R}nm \in \Theta\} \\ V^{\Theta}(p) & = & \{\bar{n} \mid n : p \in \Theta\}. \end{array}$$

We want to show that the tableau systems we present are sound and complete, i.e., that for any formula φ, $\mathcal{T}(\varphi)$ is open if, and only if, φ is satisfiable. Moreover, if $\mathcal{T}(\varphi)$ has an open branch Θ then $\mathcal{M}^{\Theta}$ is a model that satisfies φ. We present tableau calculi for $\mathcal{ML}(\langle\mathsf{sb}\rangle)$, $\mathcal{ML}(\langle\mathsf{br}\rangle)$ and $\mathcal{ML}(\langle\mathsf{sw}\rangle)$ in the next subsections.

7.1.1 *Sabotage*

Figure 12 introduces rules that, in combination with those in Figure 11, form a complete and sound tableau calculus for $\mathcal{ML}(\langle\mathsf{sb}\rangle)$. In this calculus, a formula $(n, S) : \varphi$ is understood as "φ holds at the state referred to by n in the model variant described by the set of sabotaged pairs S".

$$\dfrac{(n,S) : \Diamond\varphi}{\begin{array}{c}\dot{R}nm \\ nm\dot{\notin}S \\ (m,S) : \varphi\end{array}}\ (\Diamond)^{1} \qquad \dfrac{\begin{array}{c}(n,S) : \Box\varphi \\ \dot{R}nm \\ nm\dot{\notin}S\end{array}}{(m,S) : \varphi}\ (\Box)$$

$$\dfrac{(n,S) : \langle\mathsf{sb}\rangle\varphi}{\begin{array}{c}\dot{R}nm \\ nm\dot{\notin}S \\ (m,S \cup nm) : \varphi\end{array}}\ (\langle\mathsf{sb}\rangle)^{1} \qquad \dfrac{\begin{array}{c}(n,S) : [\mathsf{sb}]\varphi \\ \dot{R}nm \\ nm\dot{\notin}S\end{array}}{(m,S \cup nm) : \varphi}\ ([\mathsf{sb}])$$

[1] m is new.

Figure 12: Tableau rules for $\mathcal{ML}(\langle\mathsf{sb}\rangle)$.

We interpret branches of this tableau calculus with the following relation-changing function: $f : (R, S) \mapsto R \setminus S$. This means that a formula $(n, S) : \varphi$ in a branch Θ should hold in the induced model variant $\mathcal{M}^{\Theta}_{S}$ defined as $\mathcal{M}^{\Theta}_{S} = \langle W^{\Theta}, R^{\Theta}_{S}, V^{\Theta}\rangle$, where $R^{\Theta}_{S} = R^{\Theta} \setminus \bar{S}$.

The rules involve the notation $nm\dot{\notin}S$. $nm\dot{\notin}S$ specifies that the edge referred to by the pair of nominals (n, m) should not be deleted in the model variant described by S. When present as premise of a rule, this condition requires that one of the disjuncts in $nm\dot{\notin}S$ is present in the branch, which in turn means that either $n\dot{\neq}x$ or $m\dot{\neq}y$ is in the branch for all $xy \in S$.

The $(\Diamond)$ rule captures the standard meaning of the $\Diamond$ connector, but adds a new constraint that specifies that the successor has not been deleted at this point of the branch. $(\Box)$ should also take this into account. For each successor m of n in the initial model ($\dot{R}nm$), and only if the edge between n and m has not been sabotaged

$(nm\dot{\notin}S)$, φ must hold at m in the same model variant. Rule $([\mathsf{sb}])$ is similar to $(\Box)$, but φ must hold at m in the model variant where the edge nm is sabotaged. Rule $(\langle\mathsf{sb}\rangle)$ corresponds similarly to $(\Diamond)$.

We will now prove completeness and soundness of the calculus for $\mathcal{ML}(\langle\mathsf{sb}\rangle)$.

Lemma 7.1.8. *Let Θ be a saturated, open branch and φ an $\mathcal{ML}(\langle\mathsf{sb}\rangle)$-formula. If $(n, S) : \varphi \in \Theta$ then $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \varphi$.*

Proof. Let $(n, S) : \varphi \in \Theta$. Proceed by structural induction on φ.

- $\boldsymbol{\varphi = p}$: By definition, $\bar{n} \in V^{\Theta}(p)$, then $\mathcal{M}^{\Theta}, \bar{n} \models p$ and $\mathcal{M}^{\Theta}_{S}, \bar{n} \models p$.

- $\boldsymbol{\varphi = \neg p}$: By saturation of (Id), $\bar{n} : \neg p \in \Theta$. Since Θ is open, $\bar{n} : p \notin \Theta$. By definition, $\bar{n} \notin V^{\Theta}(p)$, then $\mathcal{M}^{\Theta}, \bar{n} \not\models p$ and $\mathcal{M}^{\Theta}_{S}, \bar{n} \not\models p$.

- $\boldsymbol{\varphi = \psi \wedge \chi}$ and $\boldsymbol{\varphi = \psi \vee \chi}$: Trivial by inductive hypothesis.

- $\boldsymbol{\varphi = \Diamond\psi}$: By $(\Diamond)$, Θ contains $\dot{R}nm$, $nm\dot{\notin}S$ and $(m, S) : \psi$. We want to show that $\bar{n}\bar{m} \in R^{\Theta}_{S}$. We verify the following:

  1. $\bar{n}\bar{m} \in R^{\Theta}$: this is true since $\dot{R}nm \in \Theta$.
  2. $\bar{n}\bar{m} \notin \bar{S}$: this is true since $(nm\dot{\notin}S) \in \Theta$ by Lemma 7.1.4.

  Since $\bar{n}\bar{m} \in R^{\Theta}_{S}$, and (by (Id)) $(\bar{m}, S) : \psi \in \Theta$, we have $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \Diamond\psi$.

- $\boldsymbol{\varphi = \langle\mathsf{sb}\rangle\psi}$: We need to show that $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \langle\mathsf{sb}\rangle\psi$, i.e., there exists $x \in V^{\Theta}$ s.t. $\mathcal{M}^{\Theta}_{S\cup pq}, x \models \psi$, where $\bar{p} = \bar{n}$ and $\bar{q} = x$. This can be checked considering $(\langle\mathsf{sb}\rangle)$ instead of $(\Diamond)$ as for the previous case.

- $\boldsymbol{\varphi = \Box\psi}$: We only consider states $x \in W^{\Theta}$ such that $\bar{n}x \in R^{\Theta}_{S}$. That is, there exists a, b such that $\dot{R}ab \in \Theta$ and $\bar{n}x = \bar{a}\bar{b}$, and $\bar{n}x \notin \bar{S}$. The condition of rule $(\Box)$ $(nm\dot{\notin}S)$ allows to apply it on such pair of nominals.

  By (Id), $(\bar{n}, S) : \Box\psi \in \Theta$, i.e., $(\bar{a}, S) : \Box\psi \in \Theta$, and also by $(R\sim)$, $\dot{R}\bar{a}\bar{b} \in \Theta$.

  By $(\Box)$ we have $(\bar{b}, S) : \psi \in \Theta$. Now, $\bar{b} = x$, so $\mathcal{M}^{\Theta}_{S}, x \models \psi$. Hence for all $x \in V^{\Theta}$ such that $\bar{n}x \in R^{\Theta}_{S}$, $\mathcal{M}^{\Theta}_{S}, x \models \psi$, i.e., $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \Box\psi$.

- $\boldsymbol{\varphi = [\mathsf{sb}]\psi}$: We need to show that $\mathcal{M}^{\Theta}_{S}, \bar{n} \models [\mathsf{sb}]\psi$, i.e., for all $x \in W^{\Theta}$ such that $(\bar{n}, x) \in R^{\Theta}_{S}$, $\mathcal{M}^{\Theta}_{S\cup pq}, x \models \psi$, where $\bar{p}\bar{q} = \bar{n}x$. This can be checked considering rule $([\mathsf{sb}])$ instead of $(\Box)$ as for the previous case.

$\Box$

By the previous lemma we get:

Theorem 7.1.9 (Completeness). *If $\mathcal{T}(\varphi)$ is open, then φ is satisfiable.*

We now show soundness of the calculus for $\mathcal{ML}(\langle\mathsf{sb}\rangle)$.

Lemma 7.1.10. *Let Γ be a set of satisfiable tableau formulas, and $\varphi \in \mathcal{ML}(\langle\mathsf{sb}\rangle)$. If there is a closed tableau $\mathcal{T}(\Gamma')$ for $\Gamma' = (\Gamma \cup \{\neg\varphi\})$, then φ is satisfiable.*

Proof. Let Θ be a satisfiable branch. Following Definition 7.1.2, Θ is satisfied by a model $\mathcal{M} = \langle W, R, V \rangle$ and a mapping $\sigma : \mathsf{NOM} \to W$. We write $\sigma[m \mapsto v]$ to refer to the mapping equal to σ except, perhaps, $\sigma(m) = v$.

Assume that there is a closed tableau $\mathcal{T}(\Gamma')$ such that $\Gamma' = (\Gamma \cup \{\neg\varphi\})$. We will prove Γ' unsatisfiable, by induction on the tableau structure.

- $(\bot_{atom})$: If this rule applies, then $n : a \in \Gamma'$ and $n : \neg a \in \Gamma'$, for some n, a. Then Γ' is trivially unsatisfiable.
- Common rules $(\bot_{\neq})$, $(\wedge)$, $(\vee)$, $(R\sim)$, (Id) and (ub) are easy to check.

It remains to verify, for each remaining rule, that their application to a satisfiable branch generates at least one satisfiable branch. In the present calculus, all remaining rules are non-branching.

- $(\Diamond)$: Suppose $(n, S) : \Diamond\varphi \in \mathcal{T}(\Gamma')$. We know that $(n, S) : \Diamond\varphi$ is satisfiable, then there is a model $\mathcal{M} = \langle W, R, V \rangle$, and a mapping $\sigma : \mathsf{NOM} \to W'$ s.t. $\mathcal{M}^-_{S^\sigma}, \sigma(n) \models \Diamond\varphi$. By definition of $\models$, there exists $v \in W$ s.t. $\sigma(n)v \in R \setminus S^\sigma$ and $\mathcal{M}^-_{S^\sigma}, v \models \varphi$. The $(\Diamond)$ rule generates $\dot{R}nm$, $nm\dot{\notin}S$ and $(m, S) : \varphi$, with m new in the branch. We need to check that the branch containing these three new formulas is satisfiable. That is, there exists a model and a mapping satisfying them. Let us consider the mapping $\sigma' = \sigma[m \mapsto v]$ and check that the interpretation $\mathcal{M}, \sigma'$ satisfies the new branch:
  - $\dot{R}nm$ is satisfied since $R\sigma'(n)\sigma'(m)$, i.e., $R\sigma(n)v$, holds.
  - Consider $nm\dot{\notin}S$. It suffices to check that for all $xy \in S$, $\sigma'(n)\sigma'(m) \neq \sigma'(x)\sigma'(y)$, i.e., $\sigma(n)v \neq \sigma'(x)\sigma'(y)$. But $\sigma(n)v = \sigma'(x)\sigma'(y)$ would contradict $\sigma(n)v \in R \setminus S^\sigma$.
  - $\mathcal{M}, \sigma'$ satisfies $(m, S) : \varphi$ since $\mathcal{M}^-_{S^{\sigma'}}, \sigma'(m) \models \varphi$ holds.
- $(\langle \mathsf{sb} \rangle)$: This case is similar to $(\Diamond)$, except that we need to check that the new tableau formula $(m, S \cup nm) : \varphi$ is satisfied. This is done considering the new mapping $\sigma' = \sigma[m \mapsto v]$ and observing that $\mathcal{M}^-_{S \cup nm^{\sigma'}}, \sigma'(m) \models \varphi$.
- $(\Box)$: Suppose $(n, S) : \Box\varphi$ and $\dot{R}nm$ are in Θ, and the condition $nm\dot{\notin}S$ holds. This implies that there exists $\mathcal{M} = \langle W, R, V \rangle$ and a mapping σ such that $\mathcal{M}^-_{S^\sigma}, \sigma(n) \models \Box\varphi$, and $R\sigma(n)\sigma(m)$, and there is no pair of nominals $xy \in S$ such that $nm = xy$. This means that for all $v \in W$ s.t. $\sigma(n)v \in (R \setminus S^\sigma)$, $\mathcal{M}^-_{S^\sigma}, v \models \varphi$ and there exists $v \in W$ s.t. $R\sigma(n)v$. We verify that $(m, S) : \varphi$ is satisfied by $\mathcal{M}, \sigma$. Since $\sigma(n)\sigma(m) \in (R \setminus S^\sigma)$, then $\mathcal{M}^-_{S^\sigma}, \sigma(m) \models \varphi$. Hence $(m, S) : \varphi$ is satisfied by $\mathcal{M}, \sigma$.
- $([\mathsf{sb}])$: This is similar to the $(\Box)$ case, but we have to show that $(m, S \cup nm) : \varphi$ is satisfied by $\mathcal{M}, \sigma$. This is done by observing that if $\mathcal{M}^-_{S^\sigma}, \sigma(n) \models [\mathsf{sb}]\varphi$ and $R\sigma(n)\sigma(m)$ then $\mathcal{M}^-_{S \cup nm^\sigma}, \sigma(m) \models \varphi$.

$\Box$

From the previous lemma we get the following result:

Theorem 7.1.11 (Soundness). *If φ is satisfiable, then $\mathcal{T}(\varphi)$ is open.*

Example 7.1.12 presents a satisfiable formula of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ that showcases most of the rules in the calculus.

Example 7.1.12. *A tableau for $\Diamond\Diamond\top \wedge [\mathsf{sb}]\Box\bot$ follows:*

(1)	$(n_0,\emptyset) : \Diamond\Diamond\top \wedge [\mathsf{sb}]\Box\bot$	*initial node*
(2)	$(n_0,\emptyset) : \Diamond\Diamond\top$	$(\wedge)$ *on (1)*
(3)	$(n_0,\emptyset) : [\mathsf{sb}]\Box\bot$	
(4)	$\dot{R}n_0n_1$	$(\Diamond)$ *on (2)*
(5)	$n_0n_1\dot{\notin}\emptyset$	
(6)	$(n_1,\emptyset) : \Diamond\top$	
(7)	$\dot{R}n_1n_2$	$(\Diamond)$ *on (6)*
(8)	$n_1n_2\dot{\notin}\emptyset$	
(9)	$(n_2,\emptyset) : \top$	
(10)	$(n_1,\{n_0n_1\}) : \Box\bot$	$([\mathsf{sb}])$ *on (3) and (4) with trivial condition* $n_0n_1\dot{\notin}\emptyset$
(11)	$n_0\doteq n_1 \quad\mid\quad n_0\dot{\neq}n_1$	(ub)

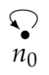
The rightmost branch soon closes since $(\Box)$ applies on (10) and (7) with condition $n_1n_2\dot{\notin}\{n_0n_1\}$ fulfilled by $n_0\dot{\neq}n_1$, and introduces $\bot$. Let us expand the leftmost branch:

(12)	$n_1\doteq n_2 \quad\mid\quad n_1\dot{\neq}n_2$	(ub)

Again the rightmost branch closes by application of $(\Box)$ with condition $n_1n_2\dot{\notin}\{n_0n_1\}$ fulfilled by $n_1\dot{\neq}n_2$. We expand the leftmost branch:

(13)	$n_0\doteq n_2 \quad\mid\quad n_0\dot{\neq}n_2$	(ub)

The right branch above closes by rule $(\bot_{\neq})$. The left branch is saturated and open, with the following induced model:



Adapting the calculus presented previously, we can obtain a sound and complete tableau method for the global sabotage operations. The corresponding $(\Diamond)$ and $(\Box)$ rules are the same ones as for its local version. Rules for $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$ in Figure 13 are direct adaptations of the rules for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

7.1.2 *Bridge*

Figure 14 presents rules for the tableau calculus corresponding to $\mathcal{ML}(\langle \mathsf{br} \rangle)$ which should be combined with the common rules of Figure 11. The main difference with rules for sabotage is that they use as prefix a set of pairs of nominals B to keep track of edges that have been added to the relation of the original model.

The interpretation function will be $f : (R, B) \mapsto R \cup B$. This means that a formula $(n, B) : \varphi$ in a branch Θ should hold in the induced model variant $\mathcal{M}^{\Theta}_{B}$ defined as

$$\dfrac{(n,S) : \langle \mathsf{gsb} \rangle \varphi}{\begin{array}{c}\dot{R}pq \\ pq \mathrel{\dot{\notin}} S \\ (n, S \cup pq) : \varphi\end{array}}\ (\langle \mathsf{gsb} \rangle)^1 \qquad \dfrac{\begin{array}{c}(n,S) : [\mathsf{gsb}]\varphi \\ \dot{R}pq \\ pq \mathrel{\dot{\notin}} S\end{array}}{(n, S \cup pq) : \varphi}\ ([\mathsf{gsb}])$$

^{1} p and q are new to the branch.

Figure 13: Tableau rules for $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$.

$$\dfrac{(n,B) : \Diamond\varphi}{\begin{array}{c|c}\dot{R}nm & nm \mathrel{\dot{\in}} B \\ (m,B) : \varphi & (m,B) : \varphi\end{array}}\ (\Diamond)^1 \qquad \dfrac{\begin{array}{c}(n,B) : \Box\varphi \\ \dot{R}nm\end{array}}{(m,B) : \varphi}\ (\Box) \qquad \dfrac{\begin{array}{c}(n,B) : \Box\varphi \\ nm \mathrel{\dot{\in}} B\end{array}}{(m,B) : \varphi}\ (\Box_2)$$

$$\dfrac{\begin{array}{c}\dot{R}nm \\ (a,B) : \varphi \\ nm \mathrel{\dot{\in}} B\end{array}}{\bot}\ (R_\bot) \qquad \dfrac{(n,B) : \langle \mathsf{br} \rangle \varphi}{\begin{array}{c}nm \mathrel{\dot{\notin}} B \\ (m, B \cup nm) : \varphi\end{array}}\ (\langle \mathsf{br} \rangle)^1 \qquad \dfrac{\begin{array}{c}(n,B) : [\mathsf{br}]\varphi \\ nm \mathrel{\dot{\notin}} B\end{array}}{(m, B \cup nm) : \varphi \mid \dot{R}nm}\ ([\mathsf{br}])^2$$

^{1} m is new.
^{2} m is already in the branch.

Figure 14: Tableau rules for $\mathcal{ML}(\langle \mathsf{br} \rangle)$.

$\mathcal{M}^\Theta_B = \langle W^\Theta, R^\Theta_B, V^\Theta \rangle$, where $R^\Theta_B = R^\Theta \cup \bar{B}$. The notation $nm \mathrel{\dot{\in}} B$ means that the edge represented by the nominals n and m is one of the edges added since the initial model in the model variant described by B. When used as a premise of a rule, the condition $nm \mathrel{\dot{\in}} B$ requires that there exists some $xy \in B$ such that $n \doteq x$ and $m \doteq y$ are present in the branch. $nm \mathrel{\dot{\notin}} B$ means that the edge (n, m) has not been added since the initial model in the variant described by B.

Some rules are more involved in this calculus. The rule $(\Diamond)$, when applied on a formula $(n, B) : \Diamond\varphi$, has to ensure that in the model variant described by B, the state referred to by the nominal n has a successor where φ holds. This model variant has a relation that is the union of the relation in the initial model and B. This is why $(\Diamond)$ is a branching rule that either chooses that the edge (n, m) belongs to the initial relation or to B.

The rule $(\Box)$ is the standard box rule for the basic modal logic. It is completed by a $(\Box_2)$ rule that ensures new edges of model variants are taken into account.

The new clash rule $(R_\bot)$ ensures that whenever some edge nm is present in a set of new edges B representing some model variant, the same edge is not present in the original model, i.e., $\dot{R}nm$ is forbidden to occur in the branch.

The rule $(\langle \mathsf{br} \rangle)$ differs from $(\Diamond)$. This is because the $\langle \mathsf{br} \rangle$ operator jumps to a state that should *not* be accessible from the current state, hence the introduction of $nm \mathrel{\dot{\notin}} B$

and $(m, B \cup nm) : \varphi$ to the branch. This last formula, together with rule $(R_\perp)$, ensures that the edge nm is not in the original model.

The rule $([\mathsf{br}])$ branches when applied to a formula $(n, B) : [\mathsf{br}]\varphi$. It decides, for every nominal m such that $nm \dot{\notin} B$, whether $(m, R \cup nm) : \varphi$ holds, or $\dot{R}nm$ holds. In the first case, together with rule $(R_\perp)$, it ensures that the edge nm is not in the original model. In the second case, it ensures the contrary, hence no bridging to m is possible and φ does not need to hold at m.

Completeness and soundness of this tableau calculus can be proved as for the sabotage tableau calculus.

Example 7.1.13 shows how the rules are used.

Example 7.1.13. *Consider the satisfiable formula $p \wedge \Diamond\neg p \wedge [\mathsf{br}]p$. In the following tableau we hide the branches that directly close by vacuity of quantification:*

$$\begin{array}{l|ll|l}
(1) & (n_0,\emptyset) : p \ \wedge \ \Diamond\neg p \ \wedge \ [\mathsf{br}]p & & \textit{initial node} \\
(2) & (n_0,\emptyset) : p & & (\wedge) \textit{ on } (1) \\
(3) & (n_0,\emptyset) : \Diamond\neg p & & \\
(4) & (n_0,\emptyset) : [\mathsf{br}]p & & \\
(5) & \dot{R}n_0n_1, (n_1,\emptyset) : \neg p & & (\Diamond) \textit{ on } (3) \\
(6) & n_0 \doteq n_1 \quad | & n_0 \dot{\neq} n_1 & (ub)
\end{array}$$

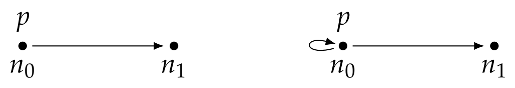
Left branch closes due to (Id) and $(\perp_{atom})$. Right branch:

$$\begin{array}{l|ll|l}
(7) & (n_1, \{n_0n_1\}) : p \quad | & \dot{R}n_0n_1 & ([\mathsf{br}]) \textit{ on } (3),\ n_1
\end{array}$$

Left branch closes by $(\perp_{atom})$ on $(n_1, \{n_0n_1\}) : p$ and $(n_1,\emptyset) : \neg p$. Right branch:

$$\begin{array}{l|ll|l}
(8) & (n_0, \{n_0n_0\}) : p \quad | & \dot{R}n_0n_0 & ([\mathsf{br}]) \textit{ on } (4),\ n_0
\end{array}$$

Both branches are open and saturated. We have the following two induced models:



As for $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$ we can adapt the calculus presented previously to get a sound and complete tableau method for the global operations. The corresponding $(\Diamond)$ and $(\Box)$ rules are the same ones as for its local version. Rules for $\mathcal{ML}(\langle\mathsf{gbr}\rangle)$ in Figure 15 are direct adaptations of the rules for $\mathcal{ML}(\langle\mathsf{br}\rangle)$.

$$\frac{(n,B) : \langle\mathsf{gbr}\rangle\varphi}{\begin{array}{c} pq \dot{\notin} B \\ (n, B \cup pq) : \varphi \end{array}} \ (\langle\mathsf{gbr}\rangle)^1 \qquad \frac{\begin{array}{c}(n,B) : [\mathsf{gbr}]\varphi \\ pq \dot{\notin} B\end{array}}{(n, B \cup pq) : \varphi \ \mid \ \dot{R}pq} \ ([\mathsf{gbr}])$$

^{1} p and q are new to the branch.

Figure 15: Tableau rules for $\mathcal{ML}(\langle\mathsf{gbr}\rangle)$.

7.1.3 *Swap*

Rules for the swap calculus are given in Figure 16, to be used in combination with the rules in Figure 11.

$$\frac{(n,S):\Diamond\varphi}{\begin{array}{c|c} \dot{R}nm & nm\dot{\in}S^{-1} \\ nm\dot{\notin}S & (m,S):\varphi \\ (m,S):\varphi & \end{array}}\,(\Diamond)^{1} \qquad \frac{\begin{array}{c}(n,S):\Box\varphi \\ \dot{R}nm \\ nm\dot{\notin}S\end{array}}{(m,S):\varphi}\,(\Box) \qquad \frac{\begin{array}{c}(n,S):\Box\varphi \\ nm\dot{\in}S^{-1}\end{array}}{(m,S):\varphi}\,(\Box_2)$$

$$\frac{\begin{array}{c}(n,S):[\mathsf{sw}]\varphi \\ \dot{R}nn\end{array}}{(n,S):\varphi}\,([\mathsf{sw}]) \qquad \frac{\begin{array}{c}(n,S):[\mathsf{sw}]\varphi \\ \dot{R}nm \\ n\dot{\neq}m \\ nm\dot{\notin}(S\cup S^{-1})\end{array}}{(m,S\cup nm):\varphi}\,([\mathsf{sw}]_2) \qquad \frac{\begin{array}{c}(n,S):[\mathsf{sw}]\varphi \\ xy\in S \\ n\doteq y\end{array}}{(x,S\backslash xy\cup yx):\varphi}\,([\mathsf{sw}]_3)$$

$$\frac{(n,S):\langle\mathsf{sw}\rangle\varphi}{\begin{array}{c|c|l} \dot{R}nn & \dot{R}nm & \bigvee_{xy\in S}(n\doteq y\wedge(x,S\backslash xy\cup yx){:}\varphi) \\ (n,S):\varphi & n\dot{\neq}m & \\ & nm\dot{\notin}(S\cup S^{-1}) & \\ & (m,S\cup nm):\varphi & \end{array}}\,(\langle\mathsf{sw}\rangle)^{1}$$

[1] m is new.

Figure 16: Tableau rules for $\mathcal{ML}(\langle\mathsf{sw}\rangle)$.

These rules have to handle the fact that swapping edges in a model can make some edges of the original model no longer usable (as when using the sabotage modality), and can make new edges usable (as with bridge). The set S that prefixes formulas of the calculus has to be understood as the pairs of states that no longer are part of the relation of the model variant. S^{-1} contains the edges that should be added to the model.

The interpretation function for this calculus is $f:(R,S)\mapsto(R\setminus S)\cup S^{-1}$. This means that a formula $(n,S):\varphi$ in a branch Θ should hold in the induced model variant $\mathcal{M}^{\Theta}_{S}$ defined as $\mathcal{M}^{\Theta}_{S}=\langle W^{\Theta},R^{\Theta}_{S},V^{\Theta}\rangle$, where $R^{\Theta}_{S}=(R^{\Theta}\ \setminus\ \bar{S})\ \cup\ \bar{S}^{-1}$.

In this calculus, S is kept irreflexive and asymmetric. Moreover, it will not contain two different pairs of nominals that refer to the same edge in the induced model. This guarantees that the names in S can be manipulated by the calculus as expected, in particular when a swapped edge must be swapped again. $nm\dot{\in}S$ means that nm is no longer present in the model variant represented by S. $nm\dot{\in}S^{-1}$ means that nm has been added to the S model variant.

Let us examine the rules. $(\Diamond)$ is a combination of the $(\Diamond)$ rules for sabotage and bridge. It satisfies the formula $(n,S):\varphi$ in a state that is either accessible through the initial relation or through a new swapped edge (as in the bridge calculus). In the case of being accessible through the initial relation, the rule ensures that the edge used has not been deleted in the current model variant (as in the sabotage calculus). The $(\Box)$ rule, as in the sabotage calculus, works with all states accessible from n in the initial model variant, except when they have been made inaccessible in the current model variant. The $(\Box_2)$ rule, as in the bridge calculus, ensures that newly accessible states receive the formula φ.

The remaining (swapping) rules deserve more careful explanation. The three rules that handle formulas of the form $[\mathsf{sw}]\varphi$ handle the case of swapping a reflexive edge, swapping an irreflexive edge that has never been swapped (nor its inverse), and swapping again an edge. $([\mathsf{sw}])$ swaps reflexive edges, for which the S set does not need to be modified since swapping a reflexive edge leaves it unchanged. $([\mathsf{sw}]_2)$ swaps irreflexive edges that have never been swapped before, i.e., usable edges (not in S) that are not in S^{-1}. This rule ensures that S is irreflexive ($n \dot{\neq} m$), asymmetric ($nm \dot{\notin} S^{-1}$) and that it does not contain two pairs of nominals that refer to the same edge in the induced model ($nm \dot{\notin} S$). Finally, $([\mathsf{sw}]_3)$ traverses and swaps around edges of S^{-1}. If $n \dot{=} y$ is in the branch and $xy \in S$ then we swap again the link yx and end up at x. Hence it removes xy from S and adds yx. This preserves the three properties of the set S (irreflexivity, asymmetry and no-redundant-names).

There is only one $(\langle\mathsf{sw}\rangle)$ rule but it handles three possibilities of satisfying a swap-diamond formula similarly to the rules for swap-box formulas. The $(\langle\mathsf{sw}\rangle)$ rule can satisfy a formula $(n, S) : \langle\mathsf{sw}\rangle\varphi$ in three possible ways. First, through a reflexive edge, having φ true at n in the same model variant. In that case S remains unchanged. Or it satisfies it by adding an irreflexive edge to the initial relation ($\dot{R}nm$, $n \dot{\neq} m$), specifying that in the model variant S it is not removed nor is a new edge added by swapping ($nm \dot{\notin} (S \cup S^{-1})$), and then satisfying φ at m in the model variant $S \cup nm$. Finally, it can satisfy the antecedent formula by swapping again a swapped edge, updating S appropriately. The meaning of the last branch of this rule is to properly maintain the set S when an edge is swapped more than once. When an edge $xy \in S$ is swapped again, we update S by removing xy and adding yx, instead of adding a new pair of nominals.

Example 7.1.14 shows the use of the tableau rules.

Example 7.1.14. *Consider the formula* $\neg p \wedge \langle\mathsf{sw}\rangle\Diamond p$.

(1)	$(n_0, \emptyset) : \neg p \wedge \langle\mathsf{sw}\rangle\Diamond p$		*initial node*
(2)	$(n_0, \emptyset) : \neg p$		$(\wedge)$ *on (1)*
(3)	$(n_0, \emptyset) : \langle\mathsf{sw}\rangle\Diamond p$		
(4)	$\dot{R}n_0n_0, (n_0, \emptyset) : \Diamond p$	$\dot{R}n_0n_1, n_0 \dot{\neq} n_1, (n_1, \{n_0n_1\}) : \Diamond p$	$(\langle\mathsf{sw}\rangle)$ *on (3)*

Let us expand the left branch:

(5a)	$\dot{R}n_0n_1, n_0n_1 \dot{\notin} \emptyset, (n_1, \emptyset) : p$		$(\Diamond)$ *on (4)*
(6a)	$n_0 \dot{=} n_1$	$n_0 \dot{\neq} n_1$	(ub)

The left branch closes by (Id) *and* $(\bot_{atom})$*, while the right branch is fully expanded and open, with the following induced model:*

$$\begin{tikzcd} \underset{n_0}{\bullet} \arrow[loop left] \arrow[r] & \underset{n_1}{\overset{p}{\bullet}} \end{tikzcd}$$

Let us go back to line (4) and expand the right branch:

(5b)	$\dot{R}n_1n_2,\ n_1n_2\dot{\notin}\{n_0n_1\}$	$n_1n_2\dot{\in}\{n_1n_0\}$	$(\Diamond)$ *on (4)*
(6b)	$(n_2, \{n_0n_1\}) : p$	$(n_2, \{n_0n_1\}) : p$	

In the right branch, by $n_1n_2\dot{\in}\{n_1n_0\}$ *we have* $n_2\dot{=}n_0$. *Then by* (Id) *and* $(\bot_{atom})$, *we have a clash. The left branch is open, and* $n_1n_2\dot{\notin}\{n_0n_1\}$ *is a notation for* $n_0\dot{\neq}n_1 \vee n_1\dot{\neq}n_2$, *with* $n_0\dot{\neq}n_1$ *already occurring in the branch (line (4), right branch).*

(7b)	$n_0\dot{=}n_2$	$n_0\dot{\neq}n_2$	(ub)

Left branch closes by (Id) *and* $(\bot_{atom})$. *Right branch:*

(8b)	$n_1\dot{=}n_2$	$n_1\dot{\neq}n_2$	(ub)

Both branch are open and saturated and produce the following induced models:

$$\begin{tikzcd} \underset{n_0}{\bullet} \arrow[r] & \overset{p}{\underset{n_1}{\bullet}} \arrow[loop right] \end{tikzcd} \qquad \begin{tikzcd} \underset{n_0}{\bullet} \arrow[r] & \underset{n_1}{\bullet} \arrow[r] & \overset{p}{\underset{n_2}{\bullet}} \end{tikzcd}$$

Now we are going to prove completeness for the $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ calculus. Soundness can be shown similarly as for sabotage.

Lemma 7.1.15. *Let* Θ *be a saturated, open branch and* φ *a* $\mathcal{ML}(\langle \mathsf{sw} \rangle)$*-formula. If* $(n, S) : \varphi \in \Theta$ *then* $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \varphi$.

Proof. Let $(n, S) : \varphi \in \Theta$, we proceed by structural induction on φ. Propositional and Boolean cases are exactly the same that for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

$\boldsymbol{\varphi = \Diamond\psi}$: We have two cases:

1. $\dot{R}nm \in \Theta$, $nm\dot{\notin}S \in \Theta$ and $(m, S) : \psi \in \Theta$. Since $\dot{R}nm \in \Theta$, we have $(\bar{n}, \bar{m}) \in R^{\Theta}$. On the other hand, since $nm\dot{\notin}S \in \Theta$ and the branch is saturated and open, by Lemma 7.1.4, $\bar{n}\bar{m} \notin \bar{S}$. Then $\bar{n}\bar{m} \in R^{\Theta}_{S}$ and (by (Id)) $(\bar{m}, S) : \psi \in \Theta$. Hence, $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \Diamond\psi$.
2. $nm\dot{\in}S^{-1} \in \Theta$ and $(m, S) : \psi \in \Theta$. From the fist sentence, by Lemma 7.1.4, we have $\bar{n}\bar{m} \in \bar{S}$, hence $\bar{n}\bar{m} \in R^{\Theta}_{S}$. With the same argument that the previous item, we have $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \Diamond\psi$.

$\boldsymbol{\varphi = \langle \mathsf{sw} \rangle \psi}$: $(\langle \mathsf{sw} \rangle)$ rule has three branches:

1. $\dot{R}nn \in \Theta$ and $(n, S) \in \Theta$. In this case $\bar{n}\bar{n} \in R^{\Theta}_{S}$, and by (Id) $(\bar{n}, S) : \psi \in \Theta$, so we have $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \langle \mathsf{sw} \rangle \psi$.
2. In the second branch, the following formulas belong to Θ: $a)$ $\dot{R}nm$, $b)$ $n\dot{\neq}m$, $c)$ $nm\dot{\notin}(S \cup S^{-1})$ and $d)$ $(m, S\ \cup nm) : \psi$. $b)$ holds since we are not in the previous case. By $a)$ and $c)$ (and Lemma 7.1.4), we have $\bar{n}\bar{m} \in R^{\Theta}_{S}$. By (Id) and $d)$, $(\bar{m}, S \cup nm) : \psi \in \Theta$. Hence, $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \langle \mathsf{sw} \rangle \psi$.
3. In the third branch, there are $x, y \in W^{\Theta}$, such that $y\dot{=}n \in \Theta$ and $(x, S \setminus xy \cup yx) \in \Theta$. Then $\bar{y}\dot{=}\bar{n} \in \Theta$ and by definition $\bar{y}\bar{x} \in R^{\Theta}_{S}$ $\otimes$. But, $(\bar{x}, S \setminus xy \cup yx) : \psi \in \Theta$, therefore $\mathcal{M}^{\Theta}_{S\setminus xy \cup yx}, \bar{x} \models \psi$. Then, since this last condition and $\otimes$, we have $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \langle \mathsf{sw} \rangle \psi$.

$\boldsymbol{\varphi = \Box\psi}$: for all $m \in W^{\Theta}$ such that $\dot{R}nm$ and $nm\dot{\notin}S \in \Theta$, we have $(m, S) : \psi \in \Theta$. Because Θ is open and saturated, by Lemma 7.1.4 it holds that $\bar{n}\bar{m} \notin \bar{S}$, which implies $\bar{n}\bar{m} \in R^{\Theta}_{S}$. Otherwise, if $nm \in S^{-1}$, then also (by definition) $\bar{n}\bar{m} \in R^{\Theta}_{S}$. In both cases, we have $(\bar{m}, S) : \psi \in \Theta$. Hence, $\mathcal{M}^{\Theta}_{S}, \bar{n} \models \Box\psi$.

$\boldsymbol{\varphi = [\mathsf{sw}]\psi}$: the reflexive case is the same as for $\Box$. If we have in Θ that $\dot{R}nm$, $n\dot{\neq}m$ and $nm\dot{\notin}(S \cup S^{-1})$, then $\bar{n}\bar{m} \in R^{\Theta}_{S}$. Also we have $(\bar{m}, S \cup nm) : \psi \in \Theta$. On the other hand, if $xy\dot{\in}S$ and $n\doteq y$ are both in Θ, (by definition) $\bar{y}\bar{x} \in R^{\Theta}_{S}$, and $(\bar{x}, S\backslash xy \cup yx){:}\psi \in \Theta$. With the three cases, we get $\mathcal{M}^{\Theta}_{S}, \bar{n} \models [\mathsf{sw}]\psi$.

$\Box$

By the previous lemma we get:

Theorem 7.1.16 (Completeness). *If $\mathcal{T}(\varphi)$ is open, then φ is satisfiable.*

The rules of a sound and complete calculus for $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$ are shown in Figure 17). Notice that ($[\mathsf{gsw}]_3$) and (the last branch produced by) ($\langle\mathsf{gsw}\rangle$) are simpler than ($[\mathsf{sw}]_3$) and ($\langle\mathsf{sw}\rangle$). This is because swapping an already swapped edge in any place is a generalization of doing it only from the evaluation state.

$$\frac{\begin{array}{c}(n,S) : [\mathsf{gsw}]\varphi \\ \dot{R}pp\end{array}}{(n,S) : \varphi}\ ([\mathsf{gsw}]) \qquad \frac{\begin{array}{c}(n,S) : [\mathsf{gsw}]\varphi \\ \dot{R}pq \\ p\dot{\neq}q \\ pq\dot{\notin}(S \cup S^{-1})\end{array}}{(n, S{\cup}pq) : \varphi}\ ([\mathsf{gsw}]_2) \qquad \frac{\begin{array}{c}(n,S) : [\mathsf{gsw}]\varphi \\ xy \in S\end{array}}{(n, S\backslash xy{\cup}yx) : \varphi}\ ([\mathsf{gsw}]_3)$$

$$\frac{(n,S) : \langle\mathsf{gsw}\rangle\varphi}{\begin{array}{c|c|c}\dot{R}pp & \dot{R}pq & \bigvee_{xy\in S}(n, S\backslash xy{\cup}yx){:}\varphi \\ (n,S) : \varphi & p\dot{\neq}q & \\ & pq\dot{\notin}(S{\cup}S^{-1}) & \\ & (n, S{\cup}pq) : \varphi & \end{array}}\ (\langle\mathsf{gsw}\rangle)^{1}$$

^{1} p and q are new to the branch.

Figure 17: Tableau rules for $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$.

All of the logics we considered can force infinite models. As a result, the tableau calculi not necessarily terminate on all inputs, given that they do not implement any kind of loop checking. However, tableau methods result helpful for different purposes. In the next sections, we will see two applications: tableaux as model building procedure, combined with model checking to obtain a terminate-and-check procedure for the satisfiability problem, and tableaux as a constructive method to compute interpolants for a hybrid version of $\mathcal{ML}(\langle\mathsf{sb}\rangle)$.

7.2 COMBINING PROCEDURES

A natural question is whether it is possible to combine these calculi into a unique calculus that would support modal logic equipped with all the relation-changing operators at once. We can easily obtain local-global combinations of calculi for operators of the same kind: $\mathcal{ML}(\langle\mathsf{sb}\rangle, \langle\mathsf{gsb}\rangle)$, $\mathcal{ML}(\langle\mathsf{br}\rangle, \langle\mathsf{gbr}\rangle)$ and $\mathcal{ML}(\langle\mathsf{sw}\rangle, \langle\mathsf{gsw}\rangle)$, by combining the corresponding rules from Section 7.1. However, further combination seems to require deep changes since every kind of relation-changing logic (sabotage, bridge, swap) requires distinct rules for the connectors $\Diamond$ and $\Box$.

Each logic has certain tableau rules that involve enforcing equality between states of the induced models. In particular, macros of the form $nm\dot{\in}S$ imply equality, and it is obvious from the semantics of the logics that equational reasoning is unavoidable. One result of this is that these logics are expressive enough to describe complex structures. It is indeed possible to find formulas that are only true in models that contain a serial, irreflexive and transitive series of states, i.e. an infinite quantity of states.

Nevertheless, we can apply a simple terminate-and-check technique (see [Areces *et al.*, 2009] for details) to the tableau procedures introduced in Section 7.1. The price to pay is that the resulting calculi are not complete. The idea is to use tableaux as a model building procedure, terminate it after a certain, predefined number of steps, and then do model checking of the input formula on the induced model. If model checking succeeds, we can answer SAT, if not the answer is NOT-KNOWN.

This procedure can be described by the Algorithm 2 for $\mathcal{ML}(\blacklozenge)$, with $\blacklozenge \in \{\langle\mathsf{sb}\rangle, \langle\mathsf{gsb}\rangle, \langle\mathsf{sw}\rangle, \langle\mathsf{gsw}\rangle, \langle\mathsf{br}\rangle, \langle\mathsf{gbr}\rangle\}$:

Algorithm 2 Incomplete SAT φ

```
procedure ISAT($\varphi$, $k \geq 0$)
    Build $\mathcal{T}(\varphi)$, a tableau with root $(n_0, \emptyset) : \varphi$, using at most $k$
    applications of the tableau rules per branch.
    if $\mathcal{T}(\varphi)$ is closed then
        return UNSAT
    else
        for $\Theta \in \mathcal{T}(\varphi)$ an open branch
        compute the corresponding $\mathcal{M}^{\Theta}$
        if $\mathcal{M}^{\Theta}, n_0 \models \varphi$ then
            return SAT
        else
            return NOT-KNOWN
        end if
    end if
end procedure
```

In Section 6.1 we proved that model checking for all logics we investigate is decidable and PSPACE-complete and hence the algorithm always terminates.

Every time the algorithm returns NOT-KNOWN, we can increase the value of the parameter k and reuse the computations performed with the smaller k. In this way, we can approximate a solution for satisfiability on this kind of logics.

7.3 HOW TO USE TABLEAUX TO COMPUTE INTERPOLANTS?

Craig's Interpolation Lemma [Craig, 1957] is a model theoretic property that has been investigated for many logics. It establishes that if $\varphi \rightarrow \psi$ is a validity, then $\varphi \rightarrow \theta$ and $\theta \rightarrow \psi$ are valid, for some θ in the common language of φ and ψ. The formula θ is called an *interpolant*. In [Fitting, 1996; Fitting, 2002; Blackburn and Marx, 2003], the interpolation lemma is proved constructively by showing how to extract an interpolant from a tableaux proof of the validity $\varphi \rightarrow \psi$. The interpolant is computed by tableau-based algorithms, adapting tableau rules to keep useful information for this computation. For $\mathcal{HL}(@,\downarrow)$, tableau formulas contain prefixes, which indicate in which point of the model the formula has to be evaluated. Together with some additional information on the polarity of formulas it is possible to systematically compute an interpolant.

Tableau formulas in Definition 7.1.1 contain prefixes which indicate in which point of the model the formula has to be evaluated, and also the current model variant. This syntactic information contained in prefixes is useful information to construct interpolants. But prefixes are not directly expressible in relation-changing modal logics. An alternative is to extend relation-changing modal logics with hybrid operators, and adapt the tableau rules of this chapter to compute interpolants. We have some preliminary results on how to obtain constructive interpolation for a hybrid version of sabotage logic ($\mathcal{HL}(@,\downarrow)+\langle \mathsf{sb} \rangle$), but it remains as future work to continue this investigation.

Part III

INFORMATION AND KNOWLEDGE

Todavía hay tiempo para imaginar cualquier cosa...

Julio Cortázar.

So far, we have investigated several dynamic operators from an abstract point of view. We analyzed in detail theoretical properties of this kind of operators, and we understood their behaviour. However, we are not only interested in an abstract perspective of relation-changing modal logics. There are several fields that use relation-changing operators to represent some particular scenarios. For instance, in *Dynamic Epistemic Logics* ($\mathcal{DEL}$) [van Ditmarsch *et al.*, 2007], this kind of operators are used to represent information change for certain agents. In *Deontic Logics*, dynamic operators can be used to reasoning about policies (operators perform modifications that stand for granting or revoking permissions [Pucella and Weissman, 2004]).

In this part, we will move to one of those concrete examples of the use of relation-changing modal logics. First, in Chapter 8 we will introduce epistemic logic, *the logic of knowledge*. We will see how to represent information and knowledge, and we will discuss what happens if we need to model change of information. We will introduce some model transformations that are appropriate for this purpose, and we will link them with the kind of operators that we investigated in the first part of the thesis.

Chapter 9 is dedicated to introduce a new relation-changing modal logic which embeds some restricted versions of dynamic epistemic logics. We will study some properties of this logics, such as invariance under bisimulation, the tree model property and expressive power. Then we will investigate in detail the computational behaviour of this logic, showing that the satisfiability problem for two different fragments is decidable. The main contribution of this chapter is the evidence we can use our experience in relation-changing modal logics in a particular setting, and we can pick a well behaved logic that suffices for our purposes.

8 Dynamic Epistemic Logics

All significant truths are private truths.
As they become public, they cease to be truth;
they become facts, or at best, part of the public character;
or at worst catchwords.

T. S. Eliot.

8.1 Going in a new direction

So far, we have studied logics in an abstract context, putting more attention on the specific properties of the languages than in their applicability to a particular problem. We investigated ways of using modal logics to represent dynamic situations, by defining operations that modify directed graphs. Hence any problem that can be represented as a problem of graph modifications can be cast in our framework. In particular, we focused on modal logics with modifiers that change the accessibility relation in a graph. We also mentioned that this kind of languages can be useful to model some specific problems, such as the strike's day in Córdoba city we discussed in Chapter 3. But there are other problems for which relation-changing modal logics could be appropriate. For instance, in [Aucher *et al.*, 2009] some relation-changing operators have been investigated as data structure modifiers. In [Kooi and Renne, 2011a], the arrow update logic (AUL) presented in Chapter 2 is introduced in an epistemic context. In the field of Epistemic Logics, many operators perform *transformations* in the model, which is why we are particularly interested on them. For instance, in [van Ditmarsch *et al.*, 2007] several operators that remove parts of the model are introduced. Relation-changing modal logics seems to be an appropriate tool for reasoning in this context. The challenge is to apply in this particular context, things we have already learned about relation-changing modal logics. For instance, we would like to exploit the good properties (such as expressivity) of the kind of logics we introduced.

The goal of this chapter is to introduce a logical approach to reason about change of information. It is a *logical* approach because the main interest is to reason about change of information, and to distinguish valid from invalid reasoning in this context. With that motivation in mind, several formalisms have been developed with different characteristics. Before discussing in detail these formal languages, we need to clarify what we mean by *information*. We consider as information something that is relative to a subject (*the agent*) who has certain perspective on the world. The knowledge of each agent is given by the information that is accessible for the agent. For this reason the concept of information change is closely related with the concept of *communication*, the process of sharing information. Communication in this context involves changing the information that the agents have, i.e., what can they observe of the world. The truth value of propositions describing the facts of the world that are independent of the agents, remains unchanged. Dynamic Epistemic Logic is the field which studies this kind of information change, as an extension of basic epistemic logic.

In this chapter we first formally introduce the different logics that are used for epistemic reasoning. Then we give some examples of situations that can be modeled using this class of logics, to see more clearly how these logics work. We discuss the connections between epistemic reasoning and the logics we investigated in previous chapters. Finally, we introduce some languages to formally represent information change, motivating the use of relation-changing modal logics in the epistemic context.

8.2 REASONING ABOUT KNOWLEDGE

Epistemic Logic is the *logic of knowledge*. Hintikka's book [Hintikka, 1962] was one of the first works that, following von Wright's ideas on modal logic [von Wright, 1951], formalized the concepts of knowledge and belief. *Possible world semantics* [Kripke, 1963] resulted fruitful to interpret diverse notions as temporal, dynamic, doxastic, deontic, and in particular, *epistemic reasoning*. In this way, knowledge and belief were formalized in a logical framework by using possible world semantics: the information that some determined agent has is given in terms of the possible worlds that are consistent with the information of that agent. Knowledge and belief are defined in terms of the accessibility of the agent to those worlds. We can say, for example, that an agent *knows* that φ is the case, if φ holds in all the worlds accessible to the agent.

Worlds in possible world semantics are nothing else than states in Kripke models (introduced in Chapter 1), and accessibility is represented by edges between states. In this chapter we slightly adapt the notation we used previously in the thesis to the most common notation in the epistemic logic field. Edges in the models are labeled by an agent symbol, which allows to model the knowledge of multiple agents, and also the knowledge that agents have of each other. We will use K and $\hat{K}$ instead of $\Box$ and $\Diamond$ in epistemic formulas, and we will also change the graphical representation of the models to the one used in epistemic logics. We will introduce this new notation with a simple example[1]. Suppose that there is an agent b who lives in Córdoba. He has a theory about the weather conditions in both Córdoba and Mendoza: it is either sunny in Córdoba (represented by the propositional symbol c) or not ($\neg c$), and it is sunny in Mendoza (represented by m) or not ($\neg m$). There are of course, four possible combinations of weather in parallel in the two cities, and each of them is a possible state of the world. This situation is modeled by the model $\mathcal{M}$ in Figure 18. An edge in epistemic models represents that an agent cannot distinguish between the two states connected by such edge. In the example, agent b lives in Córdoba, then he can distinguish if it is sunny there or not, but he cannot see the difference between the scenario in which it is sunny in Mendoza or not (in the scenario we are considering the agents have only access to the information they "see" through their window looking at the sky; no radios, newspapers or internet are available). An edge from the state w to v labeled with an agent symbol b is read as "in state w, b considers

[1] This example is a simplified version of the GLO-scenario introduced in [van Ditmarsch *et al.*, 2007].

it possible that the state in fact is v", or "agent b cannot distinguish between states w and v". Notice that the previous description refers to an equivalence relation: no agent is supposed to distinguish w from itself; if w is indistinguishable from v then so is v from w; and if b cannot distinguish w and v, and cannot distinguish v and t, then w and t are also the same from agent b. The model in Figure 18, which represents this scenario is indeed an equivalence relation. This is always the case in epistemic models, (i.e., we restrict ourselves to the class of models with equivalence relations). For historical reasons this class is known as *the class* $\mathcal{S}5$.

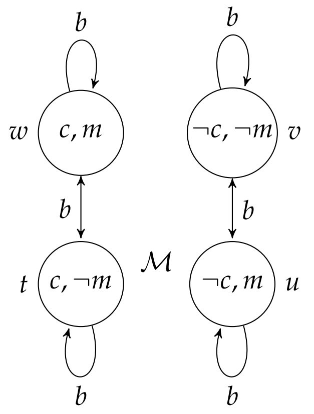


Figure 18: Epistemic model representing the weather scenario.

In this example we are modeling an external view of the scenario, without assuming any particular situation. If we are interested in checking internal properties, i.e., see which information is accessible for an agent given certain circumstances, we need to identify the state which represents that situation (the initial state or evaluation point in an epistemic pointed model). The accessible information from the initial state corresponds with a particular situation. The actual situation might be that it is sunny in Córdoba and not in Mendoza. We can see in the model that in that case the agent knows that it is sunny in Córdoba, but he is uncertain about the weather in Mendoza (he considers possible worlds where it is sunny in Mendoza and where it is raining there). We distinguish the initial state in figures with a thick contour.

We introduced a situation that can be modeled using epistemic models, that are the same structures in which we interpret epistemic formulas. We also described the intended meaning of the representation via epistemic models, in an informal way. Now, we introduce formally the epistemic language $\mathcal{EL}$, interpreted on epistemic $\mathcal{S}5$ models. The basic language for knowledge is based on a countable set of propositional symbols and a finite set of agent symbols. Atomic propositional symbols such as $p, q, r, \ldots$ describe some state of affairs (e.g., in the actual world, in a game, etc.). $\mathcal{EL}$ extends the propositional language with an unary operator for each agent. K_a is a box-like operator, with $a \in \mathsf{AGT}$, the set of agents. $K_a\varphi$ is read as "the agent a knows that φ is the case". $\hat{K}_a\varphi$ is a shorthand for $\neg K \neg\varphi$ and is pronounced as "the agent a considers it is possible that φ". Sometimes we refer to *modalities* instead of agents.

We can also introduce some notation to talk about the knowledge of a group of agents. For any group $B \subseteq \mathsf{AGT}$, we write $E_B\varphi$ and we say "everybody in B knows φ", which is defined as

$$E_B\varphi = \bigwedge_{b\in B} K_b\varphi.$$

Analogously to $\hat{K}$, $\hat{E}_B\varphi$ is a shorthand for $\neg E_B\neg\varphi$ and it is read as "at least one agent in B considers φ as a possibility".

Some examples of epistemic formulas and their meaning are:

1. $p \land \neg K_a p$: "p is true but a does not know it".
2. $\neg K_b K_c p \land \neg K_b \neg K_c p$: "agent b does not know whether agent c knows p".
3. $K_a(p \to E_B p)$: "agent a knows that if p is true, then everybody in B knows it".

We introduced the language and some informal examples which clarify the kind of information that we are able to represent with epistemic logics. We now move to the formal semantics, first defining the models in which we evaluate epistemic formulas.

As we discussed for the weather example, epistemic models are Kripke models with multiple accessibility relations, one for each agent in the language. We also mentioned that in epistemic logics, we usually focus in a particular class of models that have certain properties, which are appropriate for the corresponding reasoning. This class is $\mathcal{S}5$, i.e., the class of models such that all their accessibility relations are equivalence relations.

Definition 8.2.1 ($\mathcal{S}5$ Models). *The class of models $\mathcal{S}5$ is the class of all $\mathcal{M} = \langle W, R, V\rangle$, where R is a family of relations contained in W^2, such that for all $R' \in R$, R' is reflexive, symmetric and transitive (it is an equivalence relation).*

Given a pointed model $\mathcal{M}, w$ the semantics of the new operator is defined as

$$\mathcal{M}, w \models K_a\varphi \text{ iff } \text{ for all } v \in W \text{ s.t. } (w,v) \in R_a, \mathcal{M}, v \models \varphi.$$

The satisfiability definition establishes that an agent a knows an assertion φ in some state w of the model $\mathcal{M}$, if and only if such assertion holds in all the states that a considers possible from w. For instance, in Figure 18 we have that if it is sunny in Córdoba (i.e., w is any of the two possible worlds on the left of the figure) then $\mathcal{M}, w \models K_b c \land \neg K_b m \land \neg K_b \neg m$: *agent b knows that it is sunny in Córdoba, but does not if it is sunny in Mendoza or not*. Agent b also knows about his/her ignorance: $\mathcal{M}, w \models K_b \neg K_b m$.

As is usual in modal logics, it is interesting to know what are we able to say in the language. For instance, in epistemic logics we are interested to know how much can two epistemic states differ, without affecting the knowledge of any agent. To address this problem, we use the notion of bisimulations such as it is expected.

The logic $\mathcal{EL}$ is a generalization of the basic modal logic $\mathcal{ML}$ with multiple agents. The notion of bisimulation is exactly the same, but relating elements according to the different accessibility relations. It is easy to see that $\mathcal{EL}$ cannot distinguish bisimilar models using this notion. States that are linked by a bisimulation represent the same knowledge.

As we mentioned, epistemic logic is a classical language to talk about knowledge. Because in this thesis we are particularly interested in investigating logics to represent dynamic behaviour, it results natural to continue in this direction. In the next section, we introduce the notion of *information change*, by discussing some examples and investigating languages that are appropriate to represent it. The main goal is to find connections between this kind of languages and the relation-changing modal logics we investigate.

8.3 INFORMATION CHANGE

Dynamic epistemic logics ($\mathcal{DEL}$) are extensions of $\mathcal{EL}$ to model changing knowledge. $\mathcal{DEL}$ involves information change due to communication. The propositional information related to each state remains unchanged, but the agent's information about the states is changed. There are various examples of dynamic epistemic logics that fit with the kind of languages we are investigating. In these logics, information change is modeled with changes in the accessibility relation associated to the agents. For instance, remember that "indistinguishability" is represented in epistemic models as an edge linking two states. Hence, deleting an edge is the way in which we can differentiate two states.

8.3.1 *A Card Game Scenario*

Consider the scenario where two agents, named Ann and Bob (represented by the agent symbols a and b, respectively) have each a given card. The face side of the cards can be red or white. Both agents can only see their own card and they are ignorant about the other agent's card. There are four possibilities: both cards are white, both cards are red, Ann has a red card and Bob has a white card, or the opposite. These four possibilities are modeled by four states in a model. States are labeled with propositional symbols which indicate each situation. For instance, r_a, w_b represents the state in which Ann has a red card and Bob has a white card. Let us suppose that the initial state is r_a, r_b (both agents have a red card). All this information is represented in Figure 19 (notice that an agent cannot have a red and a white card simultaneously, it means that if a state is labeled by the symbol r_a, then $\neg w_a$ holds at such state).

States in the model are connected by edges labeled with the agent symbols a or b. An a-edge (respectively b-edge) connecting two states means that Ann (respectively Bob) cannot distinguish the two states that are connected. Each state has a reflexive edge, because no state can be distinguished from itself. R_a and R_b are equivalence relations, as we required for epistemic models. There are no a-edges between states in which Ann has different cards: she can distinguish states at the top part of the figure (she has a red card) from those at the bottom (she has a white card). Bob can distinguish states at the leftmost side of the model from those at the rightmost side. Remember this is exactly what we wanted to model, Ann and Bob only know their own cards, but are ignorant about the other card.

Suppose that Ann reveals to Bob that she has a *red card*. States where Ann has a white card disappear. This situation is represented in Figure 20. Bob is now fully

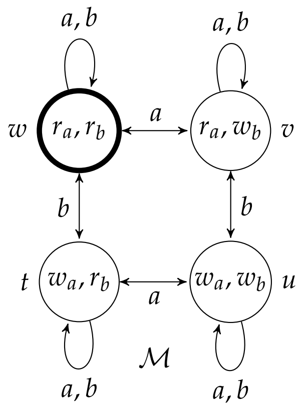


Figure 19: Epistemic model for a situation in which two agents have a red o a white card.

informed. He knows that Ann has a red card ($K_b r_a$), and they both know that they know she has a red card ($K_b(K_b r_a \wedge K_a r_a) \wedge K_a(K_b r_a \wedge K_a r_a)$), but Ann is still ignorant about Bob's card ($K_a r_b \wedge K_a r_b$). He knows he has a red card ($K_b r_b$). Notice that there is no edge labeled by b linking different states. Hence the "indistinguishability" relation for Bob is as small as possible, meaning that he has as much information as there is to have.

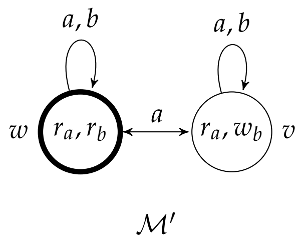


Figure 20: Epistemic model after Ann tells Bob she has a red card.

Notice also that we have not yet represented in the language the fact that Ann told Bob about her card.

Suppose now that Bob puts his card face down on a table and leaves the room. When he comes back, there are two possible situations: Ann saw his card or she did not see his card. Consider the scenario in which Ann did not see Bob's card. Bob lost some information: now he is uncertain about Ann's knowledge. Figure 21 shows this situation, in which edges for Bob appeared again. Even though Bob still knows that Ann has a red card, he is not completely informed. He does not know if Ann knows the colour of his card.

All these situations that we introduced informally can be logically described. As we can see, we started with a model modeling a scenario and we represent

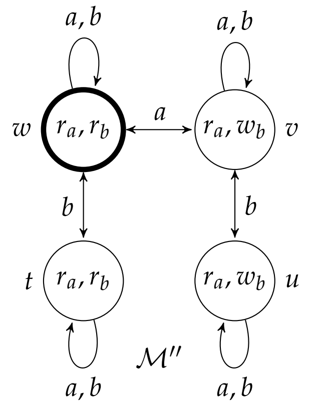


Figure 21: Ann might have seen Bob's card.

the information change by transforming the model. *Dynamic Epistemic Logics* are extensions of $\mathcal{EL}$ which allow to describe this kind of information change.

8.3.2 *Transforming Models*

Several languages have been defined to represent information change. We will discuss two main approaches: Public Announcement Logic ($\mathcal{PAL}$) and Action Model Logic ($\mathcal{AML}$). The second one is perhaps the most common approach in $\mathcal{DEL}$, and it is sometimes identified with $\mathcal{DEL}$.

$\mathcal{PAL}$ was introduced in [Plaza, 2007] (first published in 1989), as an extension of $\mathcal{EL}$ with the operator $[!\varphi]$ which communicates some common information to the agents ($\langle !\psi\rangle\varphi$ is a shorthand for $\neg[!\psi]\neg\varphi$.) The formula $[!\psi]\varphi$ is read as "after ψ is (truthfully) announced, φ is the case". It means that ψ is revealed to all the agents (the announcement is public), then φ is evaluated. Announcements are represented by removing the access to states of the model where the fact announced does not hold. We introduce the formal semantics of $\mathcal{PAL}$:

$$\mathcal{M}, w \models [!\psi]\varphi \text{ iff } \mathcal{M}, w \models \psi \text{ implies } \mathcal{M}_{|\psi}, w \models \varphi,$$

where $\mathcal{M}_{|\psi} = \langle W', R', V'\rangle$ is defined as follows:

$$\begin{array}{lcl} W' & = & \{w \in W \mid \mathcal{M}, w \models \psi\} \\ R'_a & = & R_a \cap (W' \times W') \\ V'(p) & = & V(p) \cap W'. \end{array}$$

After making an announcement, the model is transformed to a new one and evaluation of the rest of the formula continues in the new model. Agents cannot access anymore information which contradicts the announcement: the knowledge of the agents has changed. Notice that the propositional information contained in states (the valuation) does not change. The only information affected is the knowledge that the agents have of this information. This is the idea of *communication*. Let us see an example to explain how public announcements work.

Example 8.3.1. *Consider the situation modeled in Figure 19. Consider now, the formula* $[!r_a]K_b r_a$. *The formula says "after it is announced that Ann has a red card, Bob knows that Ann has a red card". The announcement* $[!r_a]$ *removes from the model* $\mathcal{M}$ *states where Ann does not have a red card. The resulting model is model* $\mathcal{M}'$ *of Figure 20: in all the remaining states Ann has a red card. All the b-edges reach states in which Ann has a red card, then* $K_b r_a$ *holds.*

The model obtained in Figure 20 is the result of deleting the states which contradict the announcement. If we consider w as the initial state, it would suffice to delete the access to the states that do not hold the announcement, such as is introduced in [van Benthem and Liu, 2007] for *Preference Upgrade*. Let us take a look to the model $\mathcal{M}'$ of Figure 22, in which we maintain the states but we remove the edges that allow the agents access to such states. In this way, we get a bisimilar model to the one in Figure 20, i.e., a model which represents the same knowledge of the agents. This seems to be an evidence that we can move to a relation-changing framework to represent informative events.

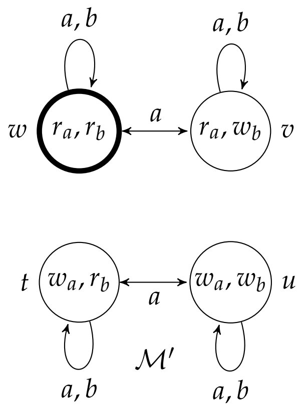


Figure 22: Ann tells Bob she has a red card, only removing edges.

The logic of public announcements is very simple and it is a good example to show the kind of languages we investigated in this thesis. Many situations can be modeled using public announcements, and it is clear that what is interesting is the notion of model transformation, such as the logics we investigated in Part II of this thesis.

$\mathcal{PAL}$ is not the only language to model informative events. Another interesting example is *Action Model Logic* ($\mathcal{AML}$) introduced in [Baltag *et al.*, 1998]. The main idea in $\mathcal{AML}$ is that events which provoke changes in the information can be modeled as the information itself. Given a situation such as the one in which Ann has a red card, we can provide a Kripke model to represent it. In epistemic models, we model information contained in the states, and what the agents are able to distinguish or not. We can use the same idea to represent information-changing events.

Let us consider again the scenario in which Bob leaves the room and when he returns he does not know if Ann saw his card or not. There are three possible situations: Ann saw Bob's card and it is red (she learns r_b), Ann saw the card and it is white (she learns w_b) or she did not see the card (she learns nothing new, we can represent this by $\top$). Ann can distinguish these particular situations, but Bob cannot. We can define a model to represent this scenario: one state for each possible proposition that Ann could learn, and b-edges between them representing that all the situations mentioned before are indistinguishable for Bob. This model will be used to update the model representing the current scenario. Models used to represent events are called *action models*. Let us introduce them formally.

Definition 8.3.2 (Action Models)**.** *Let $\mathcal{L}$ be any logical language for certain sets of propositional and agent symbols* PROP *and* AGT*, respectively. An Action Model $\mathcal{E}$ is a structure $\mathcal{E} = \langle E, \rightarrow, \mathsf{pre}\rangle$, where E is a non-empty set whose elements are called action points; for each $a \in \mathsf{AGT}$, $\rightarrow(a) \subseteq E \times E$ is an equivalence relation (we will often write $\rightarrow_a$ rather than $\rightarrow(a)$); and $\mathsf{pre} : E \rightarrow \mathcal{L}$ is a precondition function. Let e be an action point in $\mathcal{E}$, the pair $(\mathcal{E}, e)$ is a pointed action model; we usually drop parentheses and call $\mathcal{E}, e$ an pointed action model.*

Preconditions are functions which assign a formula of certain language $\mathcal{L}$ to each action point. The precondition establishes the circumstances under which the action can be executed. For instance, Ann can reveal that she has a red card, only if effectively she has a red card. Using the same criteria as for epistemic models, action models belong to the class $\mathcal{S}5$. Figure 23 shows how to represent with action models the situation in which Ann might have seen Bob's card.

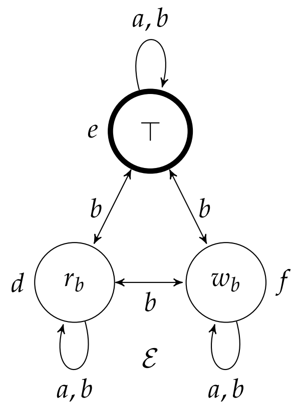


Figure 23: Action model representing that Ann might have seen Bob's card.

Action models in $\mathcal{AML}$ appear as modalities, extending the basic epistemic logic $\mathcal{EL}$.

Definition 8.3.3 (Action Model Logic Syntax)**.** *Let* PROP *be a countable, infinite set of propositional symbols and* AGT *a finite set of agent symbols. The set* FORM *of formulas of $\mathcal{AML}$ over* PROP *and* AGT *is defined as:*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid K_a\varphi \mid [\alpha]\varphi,$$

where $p \in \mathsf{PROP}$, $a \in \mathsf{AGT}$, $\varphi, \psi \in \mathsf{FORM}$ and $\alpha \in \mathsf{ACT}$. The set of actions ACT is defined as follows:

$$\mathsf{ACT} ::= \mathcal{E}, e \mid \alpha \cup \beta,$$

with $\mathcal{E}, e$ an action pointed model such that for all $d \in \mathcal{E}$, the precondition $\mathsf{pre}(d)$ is an $\mathcal{AML}$-formula constructed in a previous stage of the inductively defined hierarchy, and $\alpha, \beta \in \mathsf{ACT}$. As usual, $\langle\alpha\rangle\varphi$ is a shorthand for $\neg[\alpha]\neg\varphi$.

Action modalities indicate that some update has to be executed to the epistemic model. This is done by a model transformation called *product update*, between action models and epistemic models. Let us introduce the formal definition of the product updates, together with the full semantics for $\mathcal{AML}$.

Definition 8.3.4 (Semantics of $\mathcal{AML}$)**.** *Given an epistemic pointed model $\mathcal{M}, w$ with $\mathcal{M} = \langle W, R, V\rangle$, an action pointed model $\mathcal{E}, e$ with $\mathcal{E} = \langle E, \rightarrow, \mathsf{pre}\rangle$, and an $\mathcal{AML}$-formula φ we say that $\mathcal{M}, w$ satisfies φ (notation, $\mathcal{M}, w \models \varphi$) when*

$$\begin{array}{lll}
\mathcal{M}, w \models [\alpha]\varphi & \textit{iff} & \text{for all } \mathcal{M}', w' \text{ s.t. } \mathcal{M}, w[\![\alpha]\!]\mathcal{M}', w' \text{ we have } \mathcal{M}', w' \models \varphi \\
\mathcal{M}, w[\![\mathcal{E}, e]\!]\mathcal{M}', w' & \textit{iff} & \mathcal{M}, w \models \mathsf{pre}(e) \text{ and } \mathcal{M}', w' = (\mathcal{M} \otimes \mathcal{E}), (w, e) \\
[\![\alpha \cup \beta]\!] & = & [\![\alpha]\!] \cup [\![\beta]\!].
\end{array}$$

where $(\mathcal{M} \otimes \mathcal{E})$ is defined as $\langle W', R', V'\rangle$, with:

$$\begin{array}{lll}
W' & = & \{(v, d) \in W \times E \mid \mathcal{M}, v \models \mathsf{pre}(d)\} \\
((v, d), (u, f)) \in R'_a & \textit{iff} & (v, u) \in R_a \text{ and } d \rightarrow_a f \\
V'(p) & = & \{(v, d) \mid v \in V(p)\}.
\end{array}$$

The updated model is constructed by a product update. For each state in the original epistemic model, it is determined which actions can be executed. Preconditions in the actions do the work: an action can be executed at some state if its precondition holds at that state. The new model consists of a set of pairs (w, e), where w is an epistemic state and e is an action. (w, e) establishes that w is the resulting state after e is executed. Information in the states does not change: the valuation of a pair (w, e) follows the valuation of w. The new accessibility relation is constructed as follows. Two states which are indistinguishable to an agent should remain indistinguishable after the execution of the action. Also, two indistinguishable states in the updated model were indistinguishable in the original one. van Benthem characterizes product updates as having perfect recall, no miracles, and uniformity [van Benthem, 2001]:

- *Perfect recall:* if there is an a-edge connecting two states (w, e) and (v, d) in the updated model, then there is an a-edge between the states w and v in the original epistemic model.
- *No miracles:* if two epistemic states w and v are connected by an a-edge, then they are connected after executing the same action (i.e., (w, e) and (v, e)).
- *Uniformity:* if two states (w, e) and (v, d) are linked in the updated model by an a-edge, then two states u and t are linked by an a-edge if and only if (u, e) and (t, d) are linked (actions e and d produce the same result).

Figure 25 shows how the epistemic model $\mathcal{M}'$ is updated by the action model $\mathcal{E}$, both in Figure 24. Remember that we are modeling the scenario in which Bob is uncertain about whether Ann has seen his card or not. As expected, the obtained model after the product contains the same information that $\mathcal{M}''$ of Figure 21, which models the same situation.

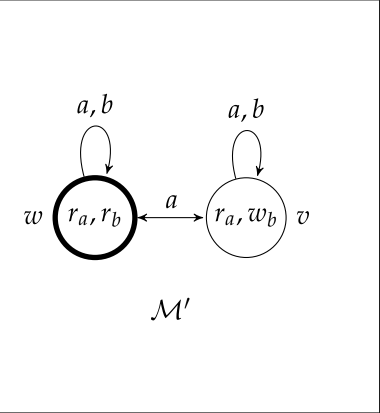
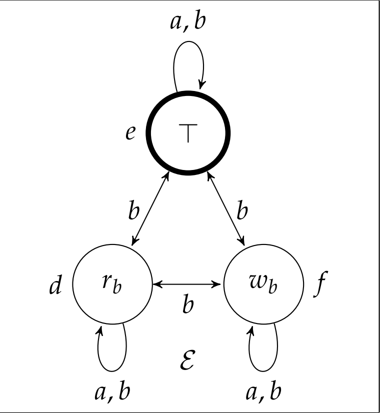
 a, b a, b w r_a, r_b a r_a, w_b v $\mathcal{M}'$	 a, b e ⊤ b b d r_b w_b f b $\mathcal{E}$ a, b a, b
Epistemic Model	Action Model

Figure 24: Epistemic model to be updated with the corresponding action model.

Notice that $\mathcal{M}'$ has two states and $\mathcal{E}$ has three states, but the product only has four. The reason is that some states do not satisfy the condition for the update of Definition 8.3.4. For instance, the pair (w, f) is thrown away because $\mathcal{M}', w \not\models \mathsf{pre}(f)$ (remember that an agent cannot have a white card and a red card at the same time). This is the case also for v and d.

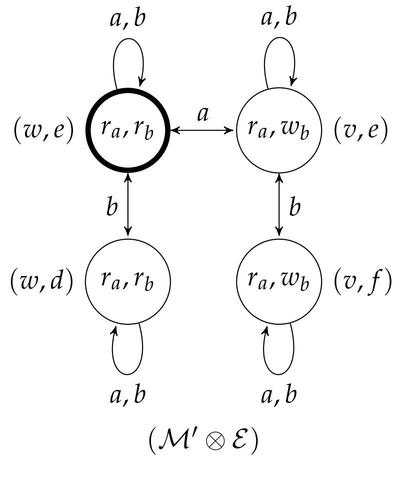


Figure 25: The model resulting of the update $\mathcal{E}$ to $\mathcal{M}'$.

$\mathcal{AML}$ is a generalization of $\mathcal{PAL}$ from a different perspective, in which action models are involved. In fact, we can represent public announcements in this framework. Consider an action model with a single state whose precondition is some formula φ. If we use this action model to update an epistemic model, the resulting product only contains the states of the epistemic model which satisfy φ. This is the effect of announcing φ: states contradicting the announcement are removed.

$\mathcal{AML}$ has been investigated in different contexts. Belief revision, mathematical puzzles, computer security and game theory are just a few applications of $\mathcal{AML}$. Results concerning the complexity and succinctness of these logics have been investigated. In [Lutz, 2006] it is shown that $\mathcal{PAL}$ has the same complexity than $\mathcal{EL}$ but it is exponentially more succinct (over the class of all models). In [French *et al.*, 2013] several other succinctness results for some extensions of $\mathcal{EL}$ (over the class of models $\mathcal{S}5$) are provided. On the other hand, NExpTime-completeness for $\mathcal{AML}$ is established in [Aucher and Schwarzentruber, 2013].

9 Applying Relation-Changing to Dynamic Epistemic Logics

The problem that we thought was a problem was, indeed, a problem,
but not the problem we thought was the problem.

Mike Smith.

Dynamic epistemic logics are particular cases of relation-changing modal logics. As we showed in the previous chapter, agent's knowledge can be represented by their accessible information, and changing the access results in information change. For this reason, communication of certain announcements or the execution of actions (which produces changes of information) can be modeled as modifications to the accessibility relation of a model. The only difference between $\mathcal{DEL}$ and the relation-changing modal logics we studied in this thesis, is that in $\mathcal{DEL}$ we give some meaning to the model (information) and formulas (knowledge), while relation-changing modal logics are more abstract, and we investigate them from a purely mathematical point of view. We are interested in what they have in common, applying the experience acquired in Part II.

We will introduce in this chapter a new relation-changing modal language which embeds $\mathcal{DEL}$. This new language is an extension of the basic modal logic $\mathcal{ML}$ with two operators: $[\![\ ]\!]$ is a modality which removes edges of a model under certain circumstances, and $\mathsf{cp}_{\bar{p}}$ is an operator that creates different copies of the model and labels each copy with exactly one of the propositional symbols p of the sequence $\bar{p}$. Clearly this language (we named it $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$) is an example of the relation-changing modal logics we investigate in this thesis. We will show that we can encode a restricted version of $\mathcal{AML}$ within $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$, and we will investigate some properties of the new language.

9.1 A Logic with Delete and Copy

We have investigated several primitives to change the accessibility relation. We defined operators to delete, swap around and add edges in a model, both locally and globally, but we have not yet combined them. In this section we introduce $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$, a language which can delete edges and create copies of a model. The main difference between $[\![\ ]\!]$ and sabotage operators is that $[\![\pi]\!]$ deletes all the edges characterized by the path expression π. Next, we introduce the formal syntax of $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$.

Definition 9.1.1 (Syntax). *Given* PROP*, an infinite and countable set of propositional symbols, and* AGT*, a finite set of agents, let us define the set* FORM *of* $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$*-formulas, together with a set* PATH *of path expressions.*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \Diamond_a \varphi \mid [\![\pi]\!]\varphi \mid cp_{\bar{p}}\varphi,$$

where $\bar{p} \in \mathsf{PROP}^+$ *does not appear in any occurrence of* cp *in* φ*,* $a \in \mathsf{AGT}$*,* $\varphi, \varphi' \in \mathsf{FORM}$*, and* $\pi \in \mathsf{PATH}$*.*

$$\mathsf{PATH} ::= a \mid \pi;\pi' \mid \varphi?,$$

where $a \in \mathsf{AGT}$*,* $\pi, \pi' \in \mathsf{PATH}$ *and* φ *is a Boolean formula.*

$[\![\pi]\!]\varphi$ is a kind of sabotage operator and its intuitive meaning is that φ holds after having deleted all edges that appear in a path that matches π. We define $[\![\pi \odot \pi']\!]\varphi$ as a shorthand for $[\![\pi]\!][\![\pi']\!]\varphi$. The operator $\mathsf{cp}_{\bar{p}}\varphi$ means that φ is true in the model obtained by replicating the initial model. We define the semantics in detail.

Definition 9.1.2 (Models). *A multimodal (or multiagent) model* $\mathcal{M}$ *is a triple* $\mathcal{M} = \langle W, R, V\rangle$*, where* W *is a non-empty set;* $R \subseteq \mathsf{AGT} \times W^2$ *is an accessibility relation; (given* $a \in \mathsf{AGT}$*, we will often write* R_a *to refer to the set* $\{(w, v) \in W^2 \mid (a, w, v) \in R\}$*); and* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *is a valuation.*

We represent a path as a sequence $w_0 a_0 w_1 a_1 \ldots w_{n-1} a_{n-1} w_n$ where w_i are states and a_i are agents. Let us now define the set $\mathcal{P}^{\mathcal{M}}_{\pi}$ of π-paths in the model $\mathcal{M}$ by induction on π. $\mathcal{P}^{\mathcal{M}}_{a}$ contains paths representing a-edges. $\mathcal{P}^{\mathcal{M}}_{\pi;\pi'}$ contains concatenations of a π-path and a π'-path. In such a concatenation, the last state w of the π-path has to be the first state of the π'-path. $\mathcal{P}^{\mathcal{M}}_{\varphi?}$ contains paths of length 0, made of one state, which satisfies φ.

Definition 9.1.3 (Paths). *Let* $\mathcal{M} = \langle W, R, V\rangle$ *a multiagent model and* $\pi \in \mathsf{PATH}$*. We define the set of* π*-paths* $\mathcal{P}^{\mathcal{M}}_{\pi}$ *of* $\mathcal{M}$ *inductively as follows:*

$$\begin{array}{lcl} \mathcal{P}^{\mathcal{M}}_{a} & = & \{wau \mid (w,u) \in R_a\} \\ \mathcal{P}^{\mathcal{M}}_{\pi;\pi'} & = & \{SwS' \mid Sw \in \mathcal{P}^{\mathcal{M}}_{\pi} \text{ and } wS' \in \mathcal{P}^{\mathcal{M}}_{\pi'}\} \\ \mathcal{P}^{\mathcal{M}}_{\varphi?} & = & \{w \mid \mathcal{M}, w \models \varphi\}. \end{array}$$

Let $a \in \mathsf{AGT}$*, we define* $edges_a(P)$ *that is the set of a-edges of the path P. Formally:*

$$edges_a(P) = \{(a, w, u) \mid wau \text{ is a subsequence of } P\}.$$

Paths are introduced to characterize modifications we can do to a model. As we did in previous chapters of the thesis, models with some modifications are called *model variants.*

Definition 9.1.4 (Model Variants). *Given a model* $\mathcal{M} = \langle W, R, V\rangle$ *and a* π*-path, we define the model variant where we delete all edges in the accessibility relation that belong to a* π*-path as*

$$\mathcal{M}_{[\![\pi]\!]} = \langle W, R_{[\![\pi]\!]}, V\rangle,$$

where

$$R_{[\![\pi]\!]} = R \setminus \bigcup_{a \in \mathsf{AGT}, P \in \mathcal{P}^{\mathcal{M}}_{\pi}} edges_a(P).$$

Let $\bar{p} \in \mathsf{PROP}^+$, we define $\mathcal{M}_{\bar{p}} = \langle W_{\bar{p}}, R_{\bar{p}}, V_{\bar{p}} \rangle$, where

$$\begin{array}{lcl} W_{\bar{p}} & = & W \times \{1, \ldots, |\bar{p}|\} \ \ (\textit{we call } w_i \textit{ to each } (w,i)) \\ R_{\bar{p}} & = & \{(a, w_i, v_j) \mid (a, w, v) \in R \wedge i, j \leq |\bar{p}|\} \\ V_{\bar{p}}(\bar{p}(i)) & = & \{w_i \in W_{\bar{p}} \mid i \leq |\bar{p}|\} \\ V_{\bar{p}}(p) & = & \{w_i \in W_{\bar{p}} \mid w \in V(p)\} \textit{ if } p \notin \bar{p}. \end{array}$$

Now we are ready to define the semantics of the operators introduced in Definition 9.1.1.

Definition 9.1.5 (Semantics). *Given a pointed model $\mathcal{M}, w$ and a formula φ we say that $\mathcal{M}, w$ satisfies φ, and write $\mathcal{M}, w \models \varphi$, when*

$$\begin{array}{lll} \mathcal{M}, w \models p & \textit{iff} & w \in V(p) \\ \mathcal{M}, w \models \neg\varphi & \textit{iff} & \mathcal{M}, w \not\models \varphi \\ \mathcal{M}, w \models \varphi \wedge \psi & \textit{iff} & \mathcal{M}, w \models \varphi \textit{ and } \mathcal{M}, w \models \psi \\ \mathcal{M}, w \models \Diamond_a \varphi & \textit{iff} & \textit{for some } v \in W \textit{ s.t. } (w,v) \in R_a,\ \mathcal{M}, v \models \varphi \\ \mathcal{M}, w \models [\![\pi]\!]\varphi & \textit{iff} & \mathcal{M}_{[\![\pi]\!]}, w \models \varphi \\ \mathcal{M}, w \models \mathsf{cp}_{\bar{p}}\varphi & \textit{iff} & \mathcal{M}_{\bar{p}}, w_1 \models \varphi. \end{array}$$

φ is satisfiable if for some pointed model $\mathcal{M}, w$ we have $\mathcal{M}, w \models \varphi$.

Figure 9.1 shows how $\mathsf{cp}_{\bar{p}}$ works, replicating the original model and tagging each copy with a particular propositional symbol of the sequence $\bar{p}$. For each state, successors of the original model are preserved, even among different copies.

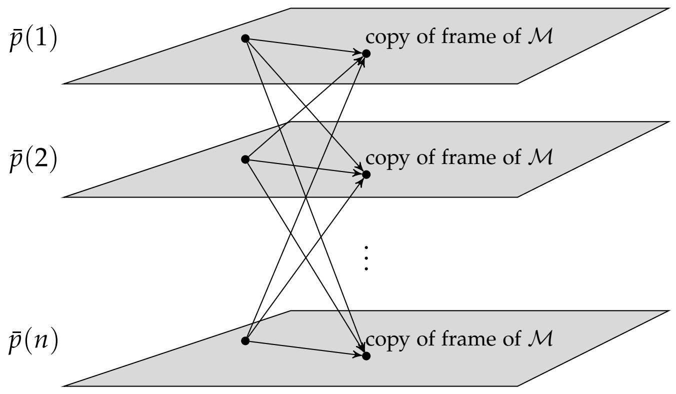
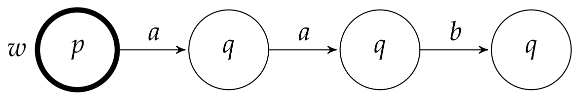

$\bar{p}(1)$
copy of frame of $\mathcal{M}$
$\bar{p}(2)$
copy of frame of $\mathcal{M}$
$\bar{p}(n)$
copy of frame of $\mathcal{M}$

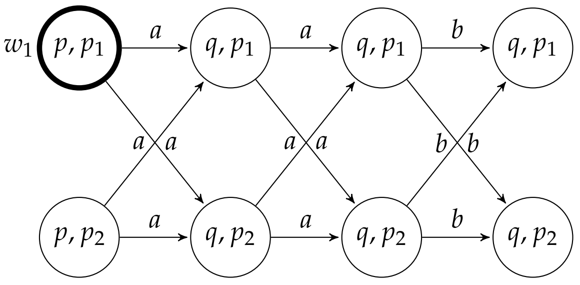
Figure 26: Semantics of copy operator.

We will discuss some examples to show the behaviour of the new operators $[\![\ ]\!]$ and cp. First, we will see a concrete example of the operator which replicates the model, and next we will see how the delete operator acts.

Example 9.1.6. *Let us consider the following model $\mathcal{M}$:*



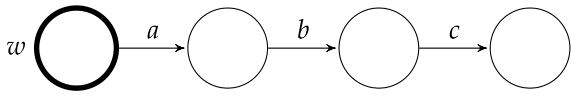
Evaluating the formula $\mathsf{cp}_{p_1 p_2}\Diamond_a\Diamond_a\Diamond_b\top$*, we first get the following model, in which we replicate the original one in two copies: one satisfying* p_1 *and another satisfying* p_2*:*



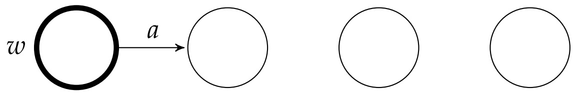
Next we have to evaluate $\Diamond_a\Diamond_a\Diamond_b\top$*. Notice that this formula is satisfied at* w *in the original model.* $\mathsf{cp}_{p_1 p_2}$ *does not introduce new information about successors: each copy of a successor in the original model is also a successor in the new model. It means that modal information remains unchanged, then* $\Diamond_a\Diamond_a\Diamond_b\top$ *holds at* w_1 *in the new model.*

Now, let us see some properties of $[\![\;]\!]$. We start by showing an example of the non-commutativity of the operator.

Example 9.1.7. *Let us consider the following model* $\mathcal{M}$*:*

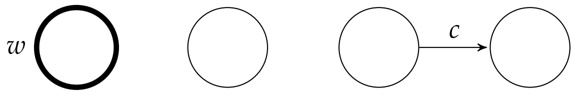


We will show that changing the order in which deletions are performed, changes the effects of the deletions. Consider the formula $[\![b;c]\!][\![a;b]\!]\Diamond_a\top$*. After evaluating* $[\![b;c]\!]$*, we get the model*



When we evaluate $[\![a;b]\!]$*, no deletions are done (the current model has no a-edges followed by b-edges), then* $\Diamond_a\top$ *holds at* w*.*

On the other hand, if we consider the formula $[\![a;b]\!][\![b;c]\!]\Diamond_a\top$*, after evaluating* $[\![a;b]\!]$ *we get:*



$[\![b;c]\!]$ *does not remove edges, and the formula* $\Diamond_a\top$ *does not hold at* w*.*

The previous example shows that $[\![\;]\!]$ is not commutative. Remember that we define $[\![\pi \odot \pi']\!]$ as $[\![\pi]\!][\![\pi']\!]$. Hence, $\odot$ is no commutative. On the other hand, notice that if $\pi = \varphi?; a; \psi?$ and $\pi' = \varphi'?; a'; \psi'?$, the order of the deletions does not matter: edges deleted by π are independent of those deleted by π' and viceversa (remember that φ, φ', ψ and ψ' are Boolean formulas). Hence, in these cases, we can assume commutativity. We will continue investigating the behaviour of the new relation-changing language. A first property we will prove is that $[\![\;]\!]$ is self-dual.

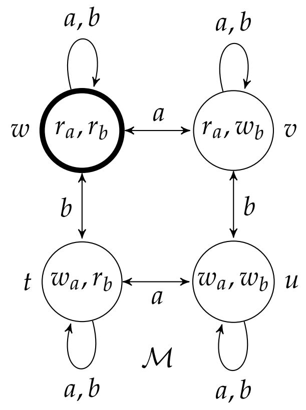
Proposition 9.1.8 (Self-duality). *Let $\mathcal{M}$ be a model, φ be an $\mathcal{ML}(cp, \llbracket\ \rrbracket)$-formula and $\pi \in \mathsf{PATH}$. Then*

$$\mathcal{M}, w \models \neg\llbracket\pi\rrbracket\neg\varphi \textit{ iff } \mathcal{M}, w \models \llbracket\pi\rrbracket\varphi.$$

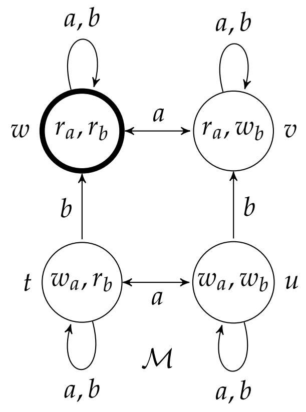
Proof. Let suppose $\mathcal{M}, w \models \neg\llbracket\pi\rrbracket\neg\varphi$. By $\models$ we have $\mathcal{M}, w \not\models \llbracket\pi\rrbracket\neg\varphi$, iff $\mathcal{M}_{\llbracket\pi\rrbracket}, w \not\models \neg\varphi$. Then, we have $\mathcal{M}_{\llbracket\pi\rrbracket}, w \models \neg\neg\varphi$, iff (by $\models$ and double negation) $\mathcal{M}, w \models \llbracket\pi\rrbracket\varphi$. □

Now let us see how $\llbracket\ \rrbracket$ can be useful to make announcements.

Example 9.1.9. *Let us consider again the example introduced in the previous chapter. Suppose we are modeling a card game scenario, and we have the following model $\mathcal{M}$, where Ann and Bob are both uncertain about the other agent's card.*



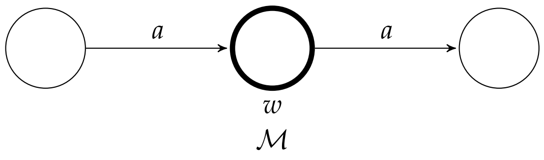
Now consider that Ann reveals to Bob she has a red card. We modeled this with a public announcement, removing all the states where Ann does not have a red card. As we mentioned in Chapter 8, this can also be modeled by removing access to such states. Evaluating the formula $\llbracket b; \neg r_a?\rrbracket K_b r_a$, we capture the same meaning that we captured in Figure 22. We are removing only b-edges, and composing it with $\neg r_a?$ we remove all the b-edges pointing to states which are not labeled by r_a. Then, according to the definition of $\models$, we are removing all the edges associated with paths in $\mathcal{P}^{\mathcal{M}}_{b;\neg r_a?}$, i.e., all the edges $(w, v) \in R_b$ such that $\mathcal{M}, v \not\models r_a$.



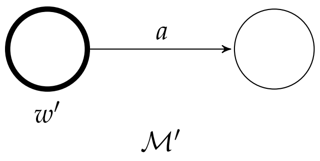
Notice that incoming b-edges to $\neg r_a$-states are removed as we did in the previous chapter, but we do not remove outgoing b-edges from $\neg r_a$-states. The resulting models are bisimilar (states where r_a does not hold are no longer accessible). After executing $[\![b; \neg r_a?]\!]$, Bob knows that Ann has a red card ($K_b r_a$).

In the first part of this thesis, we investigated in detail several relation-changing modal logics. We investigated some model properties, we introduced bisimulations, and we studied reasoning tasks. An important conclusion after investigating them, is that having relation-changing operators we can increase expressive power. Some nice computational properties are lost, in exchange, such as the tree model property and decidability. In this section we investigate in detail the new relation-changing operators. We will show that with the new definitions, which impose restrictions over which edges are modified, we can get enough expressive power for our purposes and good behaviour.

Now, we will discuss some more general properties of the whole language. A classical notion we introduce to investigate the expressive power of the languages is bisimulation. For the relation-changing modal logics we defined in Chapter 3, we needed to introduce new conditions to capture the behaviour of the new operators. Relation-changing operators increase the expressive power of the basic modal logic, and new distinctions can be drawn. For $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$, the new conditions required do not specify dynamic behaviour. Paths deleted via $[\![\ ]\!]$ traversing a particular state are characterized by the information in successors of such point, and also the predecessors. For instance, consider the formula $[\![a;a]\!]\Diamond_a\top$ and the following model:



The formula does not hold at $\mathcal{M}, w$ because we delete all the paths matching two consecutive occurrences of an a-edge. On the other hand, the formula holds in the following model:



We have that $\mathcal{M}', w' \models [\![a;a]\!]\Diamond_a\top$ because there is no path in the model matching two consecutive a-edges, and hence no edge is removed. Notice that this is the case because deletions depend on the information specified by the full path, i.e., the information that is before and after states. The new conditions for bisimilarity only have to talk about information in the past. We need to add to Definition 4.1.1, the same conditions added in [Blackburn *et al.*, 2001] for the operator $\Diamond^{-1}$:

(zig^{-1}) if $(v,w) \in R_a$ then for some v', $(v',w') \in R'_a$ and vZv';

(zag^{-1}) if $(v',w') \in R'_a$ then for some v, $(v,w) \in R_a$ and vZv'.

We use the notation $\leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])}$ when we refer to bisimulations for the language $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$.

Without loss of generality we will assume that all delete operators have the form

$$[\![\varphi_1?;a_1;\varphi_2?;a_2;\dots;a_{n-1};\varphi_n?]\!]\psi,$$

where $\varphi_i?$ are arbitrary Boolean formulas, and $a_i \in \mathsf{AGT}$ (we can introduce $\top?$ and join successive Boolean formulas by conjunction, when necessary to transform any formula into this normal form). It will be useful to prove the next lemma, which establishes that bisimulation preserves paths.

Lemma 9.1.10. *Let $\mathcal{M} = \langle W,R,V\rangle$ and $\mathcal{M}' = \langle W',R',V'\rangle$ be two models, $w \in W$ and $w' \in W'$, such that $\mathcal{M},w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',w'$. Let $\pi = \varphi_1?;a_1;\varphi_2?;a_2;\dots;a_{n-1};\varphi_n?$ be arbitrary. Then, for all $P \in \mathcal{P}^{\mathcal{M}}_{\pi}$ such that $P = w_0a_0\dots wa_i\dots w_n$, there is some $P' \in \mathcal{P}^{\mathcal{M}'}_{\pi}$, with $P' = w'_0a_0\dots w'a_i\dots w'_n$ and for all $j \in \{1,\dots,n\}$ we have $\mathcal{M},w_j \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',w'_j$.*

Proof. Given some $P \in \mathcal{P}^{\mathcal{M}}_{\pi}$, we need to find some $P' \in \mathcal{P}^{\mathcal{M}'}_{\pi}$ satisfying the lemma. Let us construct such P'.

Suppose $P = w_0a_0\dots wa_i\dots w_n$. Notice that we have the subpath wa_iw_{i+1}, which means $(w,w_{i+1}) \in R_{a_i}$. Because $\mathcal{M},w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',w'$, by (zig) there is some w'_{i+1} such that $(w',w'_{i+1}) \in R'_{a_i}$ and $\mathcal{M},w_{i+1} \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',w'_{i+1}$. For this reason, $\mathcal{M},w_{i+1} \models \psi$ if and only if $\mathcal{M}',w'_{i+1} \models \psi$, for all ψ (in particular φ_{i+1}). Then, w_{i+1} is a good choice in order to construct P'. We can repeat this process to build the subpath $w'a_iw'_{i+1}\dots w'_n$. In order to choose w_{i-1}, we can proceed in the same way but using (zig^{-1}), and repeating the process until we reach w'_1. Putting all together, we have constructed the right P'.

For the other direction use (zag) and (zag^{-1}). $\Box$

Another interesting property is that replicating bisimilar models with the operator cp, we get bisimilar models.

Lemma 9.1.11. *Let $\mathcal{M} = \langle W,R,V\rangle$ and $\mathcal{M}' = \langle W',R',V'\rangle$ be two models, $w \in W$ and $w' \in W'$. Then $\mathcal{M},w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',w'$ implies $\mathcal{M}_{\bar{p}},w_1 \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}'_{\bar{p}},w'_1$.*

Proof. We have to define a bisimulation $Z \subseteq W_{\bar{p}} \times W'_{\bar{p}}$. Because $\mathcal{M},w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',w'$, we define:

$$Z = \{(v_i,v'_i) \mid v_i \in W_{\bar{p}},\ v'_i \in W_{\bar{p}},\ i \in \{1,\dots,|\bar{p}|\} \text{ s.t. } \mathcal{M},v \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',v'\}.$$

(atomic harmony) holds because $v_iZv'_i$ if and only if v and v' satisfy (atomic harmony) in the original models, and v_i and v'_i are both labeled by the symbol $\bar{p}(i)$. For (zig), suppose we have $v_jZv'_j$ and $(v_j,u_k) \in (R_{\bar{p}})_a$. Then we know $(v,u) \in R_a$. Because $\mathcal{M},v \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',v'$, by (zig) there is some u' such that $(v',u') \in R'_a$. Hence, we have $(v'_j,u'_k) \in (R'_{\bar{p}})_a$. (zag) is straightforward. $\Box$

Now we can prove that $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$-bisimilar models satisfy the same formulas.

Theorem 9.1.12 (Invariance under bisimulation.). *For all $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$-formula φ, we have $\mathcal{M},w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}',w'$ implies $\mathcal{M},w \models \varphi$ iff $\mathcal{M}',w' \models \varphi$.*

Proof. The proof is by structural induction. Let $\mathcal{M} = \langle W, R, V\rangle$ and $\mathcal{M}' = \langle W', R', V'\rangle$, such that $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'$.

Boolean and modal cases are the same as for $\mathcal{ML}$. It remains the prove the inductive cases for $[\![\ ]\!]$ and cp.

$[\![\pi]\!]\varphi$: Suppose $\mathcal{M}, w \models [\![\pi]\!]\varphi$, then $\mathcal{M}_{[\![\pi]\!]}, w \models \varphi$, $\mathcal{M}_{[\![\pi]\!]} = \langle W, R_{[\![\pi]\!]}, V\rangle$, $R_{[\![\pi]\!]} = R \setminus \bigcup_{P\in\mathcal{P}^{\mathcal{M}}_{\pi}, a\in\mathsf{AGT}} edges_a(P)$. Because $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'$, by Lemma 9.1.10 there is $P \in \mathcal{P}^{\mathcal{M}}_{\pi}$ iff there is some $P' \in \mathcal{P}^{\mathcal{M}'}_{\pi}$. Hence $\mathcal{M}_{[\![\pi]\!]}, w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}'_{[\![\pi]\!]}, w'$, and by I.H. $\mathcal{M}'_{[\![\pi]\!]}, w' \models \varphi$. As a result, $\mathcal{M}', w' \models [\![\pi]\!]\varphi$.

$\mathsf{cp}_{\bar{p}}\varphi$: Suppose $\mathcal{M}, w \models \mathsf{cp}_{\bar{p}}\varphi$. Then we have $\mathcal{M}_{\bar{p}}, w_1 \models \varphi$. Because $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'$, by Lemma 9.1.11 we have $\mathcal{M}_{\bar{p}}, w_1 \leftrightarroweq_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}'_{\bar{p}}, w'_1$. Hence, (by I.H.) $\mathcal{M}'_{\bar{p}}, w'_1 \models \varphi$. Therefore, $\mathcal{M}', w' \models \mathsf{cp}_{\bar{p}}\varphi$.

□

The notion of bisimulation for $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$ is stronger than bisimulation for $\mathcal{ML}$, i.e., new distinctions can be drawn. However, the new conditions do not describe any dynamic behaviour, they only characterize subpaths that are back of evaluation points. Hence, we only need the same conditions as for the logic $\mathcal{ML}$ + $\Diamond^{-1}$ ($\mathcal{ML}(\Diamond^{-1})$). This establishes that properties that are not expressible in $\mathcal{ML}(\Diamond^{-1})$ are not expressible in $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$. A direct consequence is the following theorem:

Theorem 9.1.13. *The language $\mathcal{ML}(cp,[\![\ ]\!])$ has the tree model property.*

Proof. In Chapter 3 we showed the tree model property for $\mathcal{ML}$ by unraveling the original model. In [Blackburn and van Benthem, 2006] it is proved that any model $\mathcal{M}$ is bisimilar to $Unr(\mathcal{M})$. It is possible to do the same for $\mathcal{ML}(\Diamond^{-1})$ [Blackburn *et al.*, 2001]. We can encode $\Diamond^{-1}$ by adding an accessibility relation to the model, consisting in turning around existing edges and interpreting a new modality symbol on this new relation. Once we have encoded the inverse relation, we can unravel the model as for $\mathcal{ML}$ getting a bisimilar model. Hence, $\mathcal{ML}(\Diamond^{-1})$ has the tree model property. Because bisimilarity conditions for $\mathcal{ML}(\Diamond^{-1})$ and $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$ are the same, from Theorem 9.1.12 we have that $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$ has the tree model property. □

9.2 EMBEDDING ACTION MODELS IN RELATION-CHANGING

In the previous section we defined a new relation-changing language with operators to delete edges and to replicate a model. Our goal is to show that we can define $\mathcal{DEL}$ in terms of a relation-changing modal logic, such as those we investigate in Part II. To do that, we pick a restricted version of $\mathcal{AML}$ and we embed it in $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$. The constraint we impose on $\mathcal{AML}$-formulas is that preconditions of the actions in action models, are Boolean formulas.

Definition 9.2.1 (From $\mathcal{AML}$ to $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$)**.** *Let $\mathcal{E} = \langle E, \to, \mathsf{pre}\rangle$ be an action model with $E = \{\alpha_1, \ldots, \alpha_n\}$. We define the translation* Tr *from $\mathcal{AML}$ formulas to $\mathcal{ML}(cp,[\![\ ]\!])$-formulas as:*

$$\mathsf{Tr}([\mathcal{E}, \alpha_1]\varphi) = \mathsf{pre}(\alpha_1) \to cp_{p_{\alpha_1}\cdots p_{\alpha_n}}[\![\rho]\!][\![\sigma]\!]\mathsf{Tr}(\varphi),$$

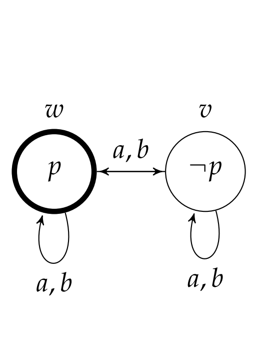
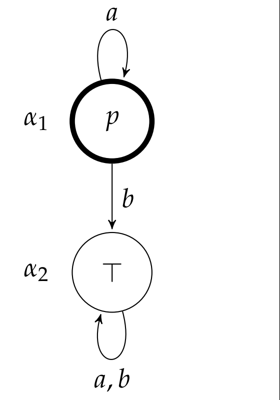
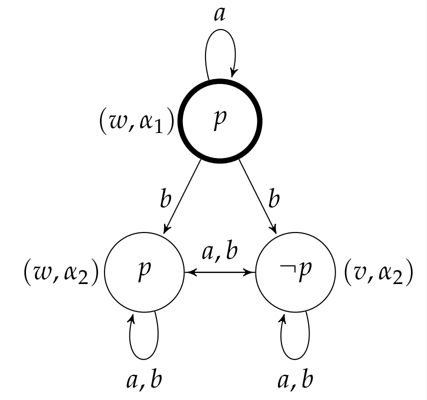
where

$$\begin{array}{rcl}\rho & \equiv & \displaystyle\bigodot_{\alpha_i\in E, a\in \mathsf{AGT}} a;(p_{\alpha_i}\wedge\neg\mathsf{pre}(\alpha_i))? \\ \sigma & \equiv & \displaystyle\bigodot_{\alpha_i,\alpha_j\in E, a\in \mathsf{AGT}} p_{\alpha_i}?;a;p_{\alpha_j}? \quad \textit{if } \alpha_i \not\to_a \alpha_j.\end{array}$$

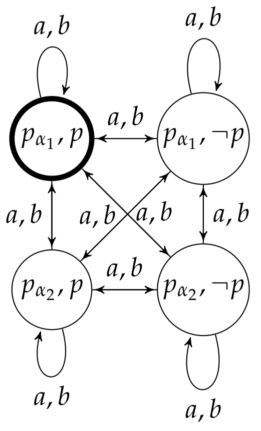
Let us discuss how the translation works, and why it is encoding action models. We use $\bigodot$ as the generalization of the shorthand $\odot$ introduced previously. Even though we showed in Example 9.1.7 that $\odot$ is not, in general, commutative, we always delete paths of size 1 (i.e., paths of the shape wav). Then, as we mentioned at the end of the same example, the order of the deletions is irrelevant. The antecedent $\mathsf{pre}(\alpha_1)$ is exactly the same clause as for model updates (considering the pointed action model $\mathcal{E},\alpha_1$ as the desired update). Then we start with the model transformation. For each action $\alpha_i \in E$, we consider a propositional symbol p_{α_i}. $\mathsf{cp}_{p_{\alpha_1}\cdots p_{\alpha_n}}$ replicates the original model as many times as actions in E. This operation generates the cartesian product $W\times E$. However, the model $\mathcal{M}\otimes\mathcal{E}$ does not consider the whole cartesian product. To cut the unwanted part of the model we introduce $[\![\rho]\!]$. In Example 9.1.9 we simulate an announcement by removing the edges pointing to states contradicting the announcement. For action models we do something similar: ρ characterizes all the edges we introduced by the previous $\mathsf{cp}_{p_{\alpha_1}\cdots p_{\alpha_n}}$ pointing to p_{α_i}-states which do not satisfy the corresponding $\mathsf{pre}(\alpha_i)$. In the same way that it is done in $\mathcal{AML}$ product updates, we remove all arrows pointing to those states. Once we have constructed the domain, it remains to restrict the obtained accessibility relation. This is done by $[\![\sigma]\!]$. Remember that $((v,d),(u,f)) \in R'_a$ in $\mathcal{M}\otimes\mathcal{E}$ if and only if $(v,u)\in R_a$ and $d\to_a f$. The first part trivially holds in the translation, because cp does not introduce edges between copies of elements that were not related in the original model. $[\![\sigma]\!]$ deletes all the a-edges (w_i,w_j) such that in the action model there is no a-edge from α_i to α_j, for all $a\in\mathsf{AGT}$.

The obtained model is not $\mathcal{M}\otimes\mathcal{E}$, but it is bisimilar according to Definition 4.1.1, which is the notion used in $\mathcal{AML}$. As a result, they represent the same information for the agents. Example 9.2.2 shows the encoding applied in a concrete scenario.

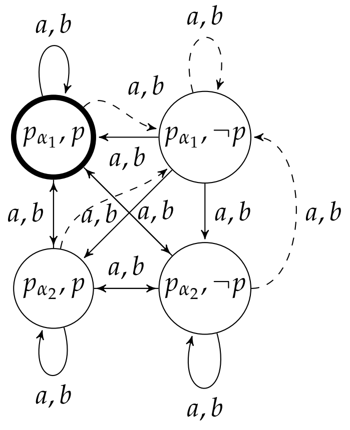
Example 9.2.2. *This is an example that shows how $\mathcal{ML}(\mathsf{cp},[\![\ ]\!])$ works encoding $\mathcal{AML}$. Below we can see an epistemic model $\mathcal{M}$ in the first column, an action model $\mathcal{E}$ in the second one, and the correspondent model after evaluating $[\mathcal{E},\alpha_1]$ at $\mathcal{M},w$.*

		 a (w, α1) p b b (w, α2) p a, b ¬p (v, α2) a, b a, b
Epistemic Model	*Action Model*	*Updated Model*

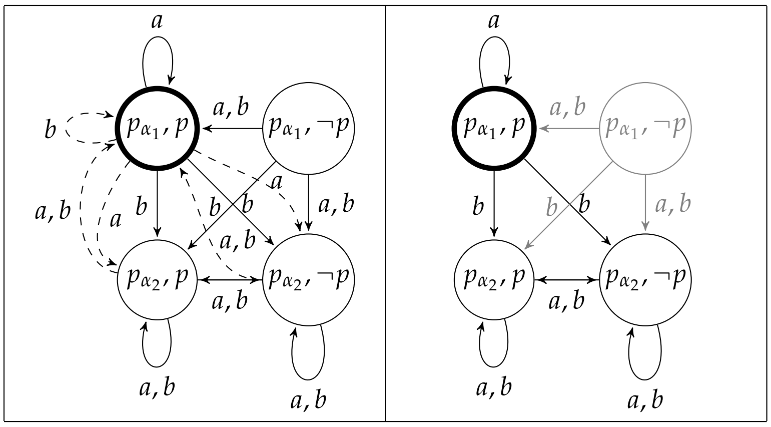
Now we will see that applying the translation Tr*, we get the same updated model (modulo* $\mathcal{AML}$*-bisimulation). We spell out each step of the transformation by evaluating each part of the translation. The first step replicates as many copies of the original epistemic model, as actions belonging to the domain of the action model. This is done via a copy operation.*



Next, evaluating $[\![\rho]\!]$ *(defined as in Definition 9.2.1), we remove all the edges pointing to states where at the same time* p_{α_1} *holds and* $\mathsf{pre}(\alpha_1)$ *doesn't hold (removed edges are represented by dashed arrows).*



Last, we need to evaluate $[\![\sigma]\!]$*. This removes those edges that have been added by the copy operation, but are not connected in the original action model. Thereby, we remove all the undesirable access, obtaining a model which is bisimilar to the updated model presented at the beginning of this example (the state labeled by* $\{p_{\alpha_1}, \neg p\}$ *is not longer accessible).*



We have seen a concrete application of the language introduced in this chapter: relation-changing modal logics to encode dynamic epistemic logics. This introduces even more evidences that some well known logics, such as dynamic epistemic logics, can be investigated in the relation-changing framework. It is clear that we cannot embed the entire logic $\mathcal{AML}$ (with action preconditions allowing modal formulas) because we do not allow modal expressions into $[\![\ ]\!]$. It remains as future work to extend $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ with modal deletions. The trade-off is the good behaviour of the considered language: for instance, we have already seen that $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ has the tree model property. In the next section we investigate more about the computational behaviour of this language. The main results are about decidability and computational complexity of the satisfiability problem for two fragments of $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$.

9.3 COMPUTATIONAL BEHAVIOUR

In this section, we start to study the satisfiability problem of two fragments of the logic $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$. First, we will consider the fragment without deletions, i.e., the basic modal logic $\mathcal{ML}$ extended with cp. We will provide a PSpace-algorithm to solve its satisfiability problem. This algorithm, together with the lower bound provided by $\mathcal{ML}$ gives us PSpace-completeness for the language. After that, we will consider $\mathcal{ML}$ extended with $[\![\ ]\!]$. We will define reduction axioms from this language to $\mathcal{ML} + \Box^{-1}$. Finally we will discuss bounds for the full language $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$.

9.3.1 *Complexity of the Fragment $\mathcal{ML}(\mathsf{cp})$*

In this subsection, we study the fragment of $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ where only the operators $\Box_a$ and $\mathsf{cp}_{\bar{p}}$ are allowed. This fragment is denoted by $\mathcal{ML}(\mathsf{cp})$.

Let Σ be an arbitrary set of sequences of propositional symbols. We define the translation Tr_Σ that maps formulas $\varphi \in \mathcal{ML}(\mathsf{cp})$ to formulas of $\mathcal{ML}$ as follows:

$$\begin{array}{lcl}
\mathsf{Tr}_\Sigma(p) & = & p \\
\mathsf{Tr}_\Sigma(\neg\varphi) & = & \neg\mathsf{Tr}_\Sigma(\varphi) \\
\mathsf{Tr}_\Sigma(\varphi \wedge \psi) & = & \mathsf{Tr}_\Sigma(\varphi) \wedge \mathsf{Tr}_\Sigma(\psi) \\
\mathsf{Tr}_\Sigma(\Box_a\varphi) & = & \Box_a(\bigwedge_{\text{where } p_i \text{ is the first propositional symbol of a sequence in } \Sigma} p_i \to \mathsf{Tr}_\Sigma(\varphi)) \\
\mathsf{Tr}_\Sigma(\mathsf{cp}_{\bar{p}}\varphi) & = & \mathsf{Tr}_{\Sigma\setminus\{\bar{p}\}}(\varphi).
\end{array}$$

We define by induction:

$$\begin{array}{lcl}\mathcal{M}_{\{\bar{p}\}\cup\Sigma} & := & (\mathcal{M}_\Sigma)_{\bar{p}} \\ \mathcal{M}_\emptyset & := & \mathcal{M}.\end{array}$$

Actually, the definition of $\mathcal{M}_\Sigma$ does not depend on the order in Σ.

Theorem 9.3.1. *For all $\varphi \in \mathcal{ML}(cp)$, let $\Sigma(\varphi)$ the set of all sequences of propositional symbols appearing in copy operators in φ. We have:*

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}_{\Sigma(\varphi)}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\varphi)}(\varphi),$$

where $w_{\Sigma(\varphi)}$ is the state corresponding to the evaluation point after $\mid\Sigma(\varphi)\mid$ consecutive cp operations are applied from the point w.

Proof. The proof is by induction on the structure of $\varphi \in \mathcal{ML}(\mathsf{cp})$.

$\boldsymbol{\varphi = p}$ **:** $\mathcal{M}, w \models p$ if and only if $\mathcal{M}_\emptyset, w \models \mathsf{Tr}_\emptyset(p)$ (by definition of $\mathcal{M}_\emptyset$ and Tr) if and only if (by definition of $\Sigma(\varphi)$) $\mathcal{M}_{\Sigma(p)}, w \models \mathsf{Tr}_{\Sigma(p)}(p)$

$\boldsymbol{\varphi = \neg\psi}$ and $\boldsymbol{\varphi = \psi \wedge \chi}$ **:** Are trivial by inductive hypothesis.

$\boldsymbol{\varphi = \Box_a\psi}$ **:** By definition of $\models$, $\mathcal{M}, w \models \Box_a\psi$ iff for all v such that $(w, v) \in R_a$, $\mathcal{M}, v \models \psi$. $\Sigma(\Box_a\psi) = \Sigma(\psi)$. By I.H., we have $\mathcal{M}_{\Sigma(\psi)}, v_{\Sigma(\psi)} \models \mathsf{Tr}_{\Sigma(\psi)}(\psi)$. This is the same as $\mathcal{M}_{\Sigma(\Box_a\psi)}, v_{\Sigma(\Box_a\psi)} \models \mathsf{Tr}_{\Sigma(\Box_a\psi)}(\psi)$. But we have $\mathcal{M}_{\Sigma(\Box_a\psi)}, w_{\Sigma(\Box_a\psi)} \models \Box_a(\bigwedge_{\text{where } p_i \text{ is the first propositional symbol of a sequence in } \Sigma(\Box_a\psi)} p_i \rightarrow \mathsf{Tr}_\Sigma(\psi))$, (because $v_{\Sigma(\Box_a\psi)}$ is an arbitrary copy of the successors of w), if and only if $\mathcal{M}_\Sigma, w_{\Sigma(\Box_a\psi)} \models \mathsf{Tr}_{\Sigma(\Box_a\psi)}(\Box_a\psi)$.

$\boldsymbol{\varphi = \mathsf{cp}_{\bar{p}}\psi}$ **:** $\mathcal{M}, w \models \mathsf{cp}_{\bar{p}}\psi$ iff $\mathcal{M}_{\bar{p}}, w_1 \models \psi$. By I.H., $(\mathcal{M}_{\bar{p}})_{\Sigma(\psi)}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\psi)}(\psi)$. Because the definition of $\mathcal{M}_\Sigma$ (which does not depend on the order of Σ), and $\bar{p} \notin \Sigma(\psi)$ (because in each occurrence of $\mathsf{cp}_{\bar{p}}$, $\bar{p}$ is fresh), we have $\mathcal{M}_{\Sigma(\psi)\cup\{\bar{p}\}}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\psi)\setminus\{\bar{p}\}}(\psi)$. Hence, by definition $\mathcal{M}_{\Sigma(\psi)\cup\{\bar{p}\}}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\psi)\cup\{\bar{p}\}}(\mathsf{cp}_{\bar{p}}\psi)$, which is the same that $\mathcal{M}_{\Sigma(\mathsf{cp}_{\bar{p}}\psi)}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\mathsf{cp}_{\bar{p}}\psi)}(\mathsf{cp}_{\bar{p}}\psi)$.

□

We will show the upper bound for the class PSpace, by providing a tableau-based algorithm which uses polynomial space. Notice that the algorithm takes as argument a $\mathcal{ML}$-formula, a set of sequences of propositional symbols and a set of formulas. At the end, we use the previous result to complete the proof.

Theorem 9.3.2. *The following problem is in* PSpace*:*

- *input: a formula $\varphi \in \mathcal{ML}$; Σ a set of sequences of propositional symbols;*
- *output: yes iff there exists a model $\mathcal{M}$ such that $\mathcal{M}_\Sigma, w_\Sigma \models \varphi$.*

Proof. We adapt the standard tableau method for $\mathcal{ML}$ (see [Goré, 1999]) in order to obtain a PSpace procedure for our problem, shown in Algorithm 3. ν is called a *modal valuation* over a set of formulas Γ if and only if $\nu \subseteq \mathsf{PROP} \cup \{\Diamond_a\psi \mid \Diamond_a\psi \text{ or } \neg\Diamond_a\psi \text{ in modal depth 1 in } \Gamma\}$ We define the relation $\models$ to say that a valuation satisfies a formula as:

$$\begin{array}{lll} \nu \models p & \text{iff} & p \in \nu \\ \nu \models \Diamond_a \varphi & \text{iff} & \Diamond_a \varphi \in \nu \\ \nu \models \neg\varphi & \text{iff} & \nu \not\models \varphi \\ \nu \models \varphi \wedge \varphi' & \text{iff} & \nu \models \varphi \text{ and } \nu \models \varphi'. \end{array}$$

A valuation c is called a *copy valuation* for a set Σ of sequences of propositional symbols, if and only if c contains exactly one propositional symbol in each sequence of Σ. Notice that, given a modal valuation ν and a copy valuation c, $\nu \cup c$ is a modal valuation.

Algorithm 3 Satisfiability for the fragment $\mathcal{ML}(\mathsf{cp})$

```
procedure SAT($\varphi, \Gamma, \Sigma$)
    choose some modal valuation $\nu$ over $\Gamma \cup \{\varphi\}$
    for all copy valuation $c$ over $\Sigma$ do
        if $\nu \cup c \not\models \bigwedge_{\gamma_i \in \Gamma} \gamma_i$ then
            return UNSAT
        end if
    end for
    if for no copy valuation $c$ we have $\nu \cup c \models \varphi \wedge \bigwedge_{\gamma_i \in \Gamma} \gamma_i$ then
        return UNSAT
    end if
    for all $\Diamond_a \psi \in \nu$ do
        if SAT($\psi, \{\neg\chi \mid \neg\Diamond_a \chi \in \nu\}, \Sigma$) =UNSAT then
            return UNSAT
        end if
    end for
    return SAT
end procedure
```

The procedure takes three arguments: a formula φ, a set of formulas Γ and a set of sequences of propositional symbols Σ. The set Γ acts as an auxiliar set, used to abstract subformulas of φ. Valuations treat formulas as propositional symbols, in which a formula belongs to a valuation if the formula has to be interpreted as true. The algorithm is the standard tableau algorithm used to check satisfiability for $\mathcal{ML}$, adapted to manage a set of sequences Σ, which represents possible copies of a model. As for $\mathcal{ML}$, the algorithm takes only polynomial space. $\square$

We will show some examples of how the algorithm works with some particular inputs.

Example 9.3.3. *Let us consider the formula $\varphi = \Diamond_a \neg p_1$ and the set $\Sigma = \{p_1\}$. Let us run SAT($\varphi, \emptyset, \Sigma$). In the first call, the valuation must be $\nu = \{\Diamond_a \neg p_1\}$. The only possible copy valuation is $c = \{p_1\}$ (because it has to make true exactly one symbol in each sequence). Intermediate checks are trivial (given that $\Gamma = \emptyset$). There is no copy valuation which contradicts $\Diamond_a \neg p_1$, then the execution of the algorithm continues by calling $SAT(\neg p_1, \emptyset, \Sigma)$. Now, $\nu = \emptyset$, and $c = \{p_1\}$, then the algorithm fails checking $\nu \cup c \models \neg p_1$ (because c contains p_1). Therefore the algorithm returns* UNSAT.

Example 9.3.4. *Now, let us consider the same formula, but with $\Sigma = \{p_1 p_2\}$. Again, we have one unique sequence but composed now by two propositional symbols. Hence, we run $SAT(\varphi, \emptyset, \Sigma)$. In the first call, the valuation must be $v = \{\Diamond_a \neg p_1\}$. In the second step, we have two possible copy valuations: $c = \{p_1\}$ and $c = \{p_2\}$ We have seen in the previous example what happens if we consider $c = \{p_1\}$, then let us choose the other option. In the recursive call $SAT(\neg p_1, \emptyset, \Sigma)$, we have $v \cup c \models \neg p_1$, with $v = \emptyset$ and $c = \{p_2\}$. With this input, the algorithm reaches its last instruction and returns* SAT.

From previous result, we can state the complexity of the satisfiability problem for the fragment $\mathcal{ML}(\mathsf{cp})$:

Theorem 9.3.5. *Deciding whether a formula in $\mathcal{ML}(cp)$ is satisfiable is* PSPACE*-complete.*

Proof. PSPACE-hardness follows from PSPACE-completeness of the satisfiability problem for $\mathcal{ML}$. In order to prove completeness, we can use Theorem 9.3.2, testing whether there exists a model $\mathcal{M}$ such that $\mathcal{M}_{\Sigma(\varphi)}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\varphi)}(\varphi)$. This can be done (by Theorem 9.3.1) by invoking

$$\mathsf{SAT}(\mathsf{Tr}_{\Sigma(\varphi)}(\varphi) \wedge \textstyle\bigwedge_{p_i, \text{ where } p_i \text{ is a first atomic proposition of a sequence in } \Sigma(\varphi)} p_i, \emptyset, \Sigma(\varphi)).$$

□

We will continue investigating the complexity of the satisfiability problem for $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$ in the next section. We will not get an upper bound for this, but we will prove that the problem is decidable.

9.3.2 *Complexity of the Fragment $\mathcal{ML}([\![\,]\!])$*

We will show that the fragment without the copy operator ($\mathcal{ML}([\![\,]\!])$) can be translated into basic modal logic with past operator $\Diamond^{-1}$ (we notate this logic as $\mathcal{ML}(\Diamond^{-1})$).

Without loss of generality we will assume that all delete operators have the form

$$[\![\varphi_1?; a_1; \varphi_2?; a_2; \ldots; a_{n-1}; \varphi_n?]\!]\psi,$$

where $\varphi_i?$ are arbitrary Boolean formulas, and $a_i \in \mathsf{AGT}$. We will introduce reduction axioms to get a $\mathcal{ML}(\Diamond^{-1})$-formula. Notice that the macros we define below (del_i^{π}'s) introduce $\Diamond^{-1}$ operators. Because we will use del_i^{π}'s in reduction axioms, intermediate steps introduce $\Diamond^{-1}$. For this reason reduction axioms are from $\mathcal{ML}([\![\,]\!], \Diamond^{-1})$-formulas to $\mathcal{ML}(\Diamond^{-1})$-formulas, with $\mathcal{ML}([\![\,]\!], \Diamond^{-1})$ the language $\mathcal{ML}([\![\,]\!])$ extended with $\Diamond^{-1}$. We will then conclude that for any $\mathcal{ML}([\![\,]\!])$-formula, we can get an equivalent $\mathcal{ML}(\Diamond^{-1})$-formula.

First, let us define the macros $\Diamond_{i,j}$ and $\Diamond_{i,j}^{-1}$, for a fixed $\pi = \varphi_1?; a_1; \ldots; a_{n-1}; \varphi_n$.

$$\Diamond_{i,j} = \begin{cases} \top & j < i \\ \Diamond_{a_i} \varphi_{i+1} & i = j \\ \Diamond_{a_i}(\varphi_{i+1} \wedge \Diamond_{i+1,j}) & i < j \end{cases} \qquad \Bigg| \qquad \Diamond_{i,j}^{-1} = \begin{cases} \top & j < i \\ \Diamond_{a_i}^{-1} \varphi_i & i = j \\ \Diamond_{a_j}^{-1}(\Diamond_{i,j-1}^{-1} \wedge \varphi_j) & i < j \end{cases}$$

Then we define the formula del_i^{π} as follows:

$$del_i^{\pi} = \Diamond_{1,i-1}^{-1} \wedge \varphi_i \wedge \Diamond_{i,n-1}.$$

Informally del_i^π means "the current world is at position i in a path that matches $\pi = \varphi_1?; a_1; \varphi_2?; a_2; \ldots; a_{n-1}; \varphi_n?$ and that is going to be deleted".

For instance, formulas $del_1^\pi, \ldots, del_n^\pi$ are defined as:

$$\begin{array}{lcl}
del_1^\pi & = & \varphi_1 \wedge (\Diamond_{a_1}\varphi_2 \wedge (\Diamond_{a_2}\varphi_3 \ldots \wedge \Diamond_{a_{n-2}}(\varphi_{n-1} \wedge \Diamond_{a_{n-1}}\varphi_n)\ldots)) \\
del_2^\pi & = & \Diamond_{a_1}^{-1}\varphi_1 \wedge \varphi_2 \wedge (\Diamond_{a_2}\varphi_3 \ldots \wedge \Diamond_{a_{n-2}}(\varphi_{n-1} \wedge \Diamond_{a_{n-1}}\varphi_n)\ldots) \\
 & \ldots & \\
del_{n-1}^\pi & = & \Diamond_{a_{n-2}}^{-1}(\Diamond_{a_{n-3}}^{-1}(\ldots(\Diamond_{a_1}^{-1}\varphi_1 \wedge \varphi_2) \wedge \varphi_3)\ldots) \wedge \varphi_{n-1} \wedge \Diamond_{a_{n-1}}\varphi_n \\
del_n^\pi & = & \Diamond_{a_{n-1}}^{-1}(\Diamond_{a_{n-2}}^{-1}(\ldots(\Diamond_{a_1}^{-1}\varphi_1 \wedge \varphi_2) \wedge \varphi_3 \ldots) \wedge \varphi_{n-1}) \wedge \varphi_n
\end{array}$$

Next lemma clarifies the meaning of del_i:

Lemma 9.3.6. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, $w \in W$ and $\pi = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_n?$ a path expression. Let i be such that $0 \leq i \leq n$, then*

$$\mathcal{M}, w \models del_i^\pi \text{ iff there is some } P \in \mathcal{P}_\pi^\mathcal{M} \text{ s.t. } P = w_1 a_1 w_2 \ldots w_n,\ w_i = w$$

and for all $w_j \in P$ we have $\mathcal{M}, w_j \models \varphi_j$.

Proof. The proof is by induction on the length of π:

$\pi = \varphi_1?$: $\mathcal{M}, w \models del_1^\pi$ if and only if $\mathcal{M}, w \models \varphi_1$ (by definition of del_i^π). But $\mathcal{P}_{\varphi_1?}^\mathcal{M} = \{v \mid \mathcal{M}, v \models \varphi_1\}$ (all the paths are singletons satisfying φ_1), then $w \in \mathcal{P}_{\varphi_1?}^\mathcal{M}$.

$\pi = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_n?$: Suppose $\mathcal{M}, w \models del_i^\pi$. By definition of del^π, we have $\mathcal{M}, w \models \Diamond_{1,i-1}^{-1} \wedge \varphi_i \wedge \Diamond_{i,n-1}$. Now, we know:

1. $\mathcal{M}, w \models \varphi_i$.

2. $\mathcal{M}, w \models \Diamond_{1,i-1}^{-1}$, then by definition of $\Diamond_{i,j}^{-1}$ we have $\mathcal{M}, w \models \Diamond_{a_{i-1}}^{-1}(\Diamond_{1,i-2}^{-1} \wedge \varphi_{i-1})$. By definition of $\models$, there is some $v \in W$ such that $(v, w) \in R_{a_{i-1}}$ and $\mathcal{M}, v \models \Diamond_{1,i-2}^{-1} \wedge \varphi_{i-1}$. Let us define $\pi_1 = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_{i-1}?$. Then, by definition of del_i^π, we have $\mathcal{M}, v \models del_{i-1}^{\pi_1}$, and by I.H., there is a path $P_1 \in \mathcal{P}_{\pi_1}^\mathcal{M}$ such that $P_1 = w_1 a_1 \ldots w_{i-1}$, with $w_{i-1} = v$ and for all $w_j \in P_1$, $\mathcal{M}, w_j \models \varphi_j$ $(0 \leq j \leq i-1)$.

3. $\mathcal{M}, w \models \Diamond_{i,n-1}$, then by definition of $\Diamond_{i,j}$ we have $\mathcal{M}, w \models \Diamond_{a_i}(\varphi_{i+1} \wedge \Diamond_{i+1,n-1})$. By definition of $\models$, there is some $t \in W$ such that $(w, t) \in R_{a_i}$ and $\mathcal{M}, t \models \varphi_{i+1} \wedge \Diamond_{i+1,n-1}$. Let us define $\pi_2 = \varphi_{i+1}?; a_{i+1}; \ldots; \varphi_n?$. Then, by definition of del_i^π, we have $\mathcal{M}, t \models del_{i+1}^{\pi_2}$, and by I.H., there is a path $P_2 \in \mathcal{P}_{\pi_2}^\mathcal{M}$ such that $P_2 = w_{i+1} a_{i+1} \ldots w_n$, with $w_{i+1} = t$ and for all $w_j \in P_2$, $\mathcal{M}, w_j \models \varphi_j$ $(i+1 \leq j \leq n)$.

Notice that $\pi = \pi_1; a_{i-1}; \varphi_i?; a_i; \pi_2$. It remains to choose $P = P_1 a_{i-1} w_i a_i P_2$ and we have what we wanted.

□

Next, we introduce the reduction axioms which transform $\mathcal{ML}(\llbracket\ \rrbracket, \Diamond^{-1})$-formulas into $\mathcal{ML}(\Diamond^{-1})$-formulas.

Definition 9.3.7. *Let $\varphi = [\![\pi]\!]\theta$ be an $\mathcal{ML}([\![\,]\!], \Diamond^{-1})$-formula, with $\pi = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_n?$. We define the formula $\mathsf{Tr}(\varphi)$ as the $\mathcal{ML}(\Diamond^{-1})$-formula resulting of apply repeatedly the following reduction axioms to the formula φ (we will assume that $\Diamond_a\psi$ is written as $\neg\Box_a\neg\psi$, and similarly by $\Diamond^{-1}$).*

$$\begin{array}{lll}
(1) & [\![\pi]\!]p & \leftrightarrow \quad p, \;\; p \in \mathsf{PROP} \\
(2) & [\![\pi]\!]\neg\psi & \leftrightarrow \quad \neg[\![\pi]\!]\psi \\
(3) & [\![\pi]\!](\psi \wedge \psi') & \leftrightarrow \quad ([\![\pi]\!]\psi \wedge [\![\pi]\!]\psi') \\
(4) & [\![\pi]\!]\Box_a\psi & \leftrightarrow \quad \Box_a[\![\pi]\!]\psi, \;\; \textit{if } a \notin \pi \\
(5) & [\![\pi]\!]\Box_a^{-1}\psi & \leftrightarrow \quad \Box_a^{-1}[\![\pi]\!]\psi, \;\; \textit{if } a \notin \pi \\
(6) & [\![\pi]\!]\Box_a\psi & \leftrightarrow \quad (\bigwedge_{i\in\{1,\ldots,n-1 \mid a_i = a\}} \neg del_i^{\pi} \to \Box_{a_i}[\![\pi]\!]\psi) \wedge \\
 & & \qquad (\bigwedge_{i\in\{1,\ldots,n-1 \mid a_i = a\}} (del_i^{\pi} \to \Box_{a_i}(del_{i+1}^{\pi} \vee [\![\pi]\!]\psi))) \\
(7) & [\![\pi]\!]\Box_a^{-1}\varphi & \leftrightarrow \quad (\bigwedge_{i\in\{1,\ldots,n-1 \mid a_i = a\}} \neg del_i^{\pi} \to \Box_{a_i}^{-1}[\![\pi]\!]\psi) \wedge \\
 & & \qquad (\bigwedge_{i\in\{1,\ldots,n-1 \mid a_i = a\}} (del_i^{\pi} \to \Box_{a_i}^{-1}(del_{i-1}^{\pi} \vee [\![\pi]\!]\psi))).
\end{array}$$

Notice that the resulting formula only contains $\Box_a$ and $\Box_a^{-1}$, and does not contain $[\![\,]\!]$. We will prove that the reduction preserves equivalence, discussing each reduction axiom separately. First, we will show that reduction axiom (1) preserves equivalence.

Lemma 9.3.8. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, $w \in W$. Let $p \in \mathsf{PROP}$ be a propositional symbol and $\pi \in \mathsf{PATH}$ arbitrary, then*

$$\mathcal{M}, w \models [\![\pi]\!]p \textit{ iff } \mathcal{M}, w \models p.$$

Proof. Suppose $\mathcal{M}, w \models [\![\pi]\!]p$. By definition of $\models$, we have $\mathcal{M}_{[\![\pi]\!]}, w \models p$. Because $[\![\pi]\!]$ keeps the same valuation in the model variant, $w \in V(p)$. Then (by $\models$), $\mathcal{M}, w \models p$. $\Box$

Next, we will see the distributivity of $[\![\,]\!]$ with respect to $\wedge$.

Lemma 9.3.9. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, ψ, ψ' be two $\mathcal{ML}([\![\,]\!], \Diamond^{-1})$-formulas and $\pi \in \mathsf{PATH}$ such that $\pi = \varphi_1?; a_i; \varphi_2?; \ldots; a_{n-1}; \varphi_n?$. Then*

$$\mathcal{M}, w \models [\![\pi]\!](\psi \wedge \psi') \textit{ iff } \mathcal{M}, w \models [\![\pi]\!]\psi \wedge [\![\pi]\!]\psi'.$$

Proof. Suppose $\mathcal{M}, w \models [\![\pi]\!](\psi \wedge \psi')$. Then, by definition of $\models$, $\mathcal{M}_{[\![\pi]\!]}, w \models (\psi \wedge \psi')$, which means $\mathcal{M}_{[\![\pi]\!]}, w \models \psi$ and $\mathcal{M}_{[\![\pi]\!]}, w \models \psi'$. Applying again definition of $\models$, we have $\mathcal{M}, w \models [\![\pi]\!]\psi$ and $\mathcal{M}, w \models [\![\pi]\!]\psi'$, iff $\mathcal{M}, w \models [\![\pi]\!]\psi \wedge [\![\pi]\!]\psi'$. $\Box$

Now we prove that in certain cases, $[\![\,]\!]$ and $\Box_a$ commute.

Lemma 9.3.10. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, ψ be a $\mathcal{ML}([\![\,]\!], \Diamond^{-1})$-formula and $\pi \in \mathsf{PATH}$ such that $\pi = \varphi_1?; a_i; \varphi_2?; \ldots; a_{n-1}; \varphi_n?$. If $a_i \notin \pi$, then*

$$\mathcal{M}, w \models [\![\pi]\!]\Box_{a_i}\psi \textit{ iff } \mathcal{M}, w \models \Box_{a_i}[\![\pi]\!]\psi.$$

Proof. Suppose $\mathcal{M}, w \models [\![\pi]\!]\Box_{a_i}\psi$. Applying definition of $\models$ twice, we have that for all v such that $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$. We assume $a_i \notin \pi$, then $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$ iff $(w, v) \in R_{a_i}$, then we have for all v such that $(w, v) \in R_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$, iff for all v such that $(w, v) \in R_{a_i}$, $\mathcal{M}, v \models [\![\pi]\!]\psi$. Hence by $\models$, $\mathcal{M}, w \models \Box_{a_i}[\![\pi]\!]\psi$. $\Box$

Finally, we prove the equivalence preservation of axiom (6). A property similar to Lemma 9.3.10 holds for $\Box_{a_i}^{-1}$ too.

Theorem 9.3.11. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, $w \in W$, and let $[\![\pi]\!]\Box_{a_i}\psi$ be an $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$-formula with $\pi = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_n?$, such that $a_i \in \pi$. Then*

$$\mathcal{M}, w \models [\![\pi]\!]\Box_{a_i}\psi \text{ iff } \mathcal{M}, w \models \delta \wedge \delta'$$

where

$$\begin{array}{ll} \delta & = \bigwedge_{k \in \{1,\ldots,n-1 \mid a_k = a_i\}} \neg del^{\pi}_k \to \Box_{a_k}[\![\pi]\!]\psi \\ \delta' & = \bigwedge_{k \in \{1,\ldots,n-1 \mid a_k = a_i\}} (del^{\pi}_k \to \Box_{a_k}(del^{\pi}_{k+1} \vee [\![\pi]\!]\psi)). \end{array}$$

Proof. Let us suppose that $\mathcal{M}, w \models [\![\pi]\!]\Box_{a_i}\psi$. Then, by definition of $\models$, we have that for all $v \in W$ such that $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$. We will check the two conjuncts δ and δ' separately (for the other direction of the iff, we can assume the two conjuncts together and use the same steps):
1. Suppose $\mathcal{M}, w \models \bigwedge_{k \in \{1,\ldots,n-1 \mid a_k = a_i\}} \neg del^{\pi}_k$. By definition of $\models$, we have $\mathcal{M}, w \not\models \bigvee_{k \in \{1,\ldots,n-1 \mid a_k = a_i\}} del^{\pi}_k$. It means that there is no $P \in \mathcal{P}^{\mathcal{M}}_{\pi}$ satisfying Lemma 9.3.6, such that $w \in P$, hence no deletions have been done traversing w. Then for all $v \in W$, $(w, v) \in R_{a_i}$ iff $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$. Because we have for all $v \in W$ such that $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$, then for all $v \in W$ such that $(w, v) \in R_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$. Therefore, we have for all $v \in W$ such that $(w, v) \in R_{a_i}$, $\mathcal{M}, v \models [\![\pi]\!]\psi$, then (by $\models$) $\mathcal{M}, w \models \Box_{a_i}[\![\pi]\!]\psi$.
2. Suppose now for some arbitrary k, $\mathcal{M}, w \models del^{\pi}_k$, where $k \in \{1, \ldots, n-1 \mid a_k = a_i\}$. By Lemma 9.3.6 it means that there is a path traversing w that has been deleted. We also know $\mathcal{M}_{[\![\pi]\!]}, w \models \Box_{a_k}\psi$ by assumption and $k = i$, then for all $v \in W$ such that $(w, v) \in (R_{[\![\pi]\!]})_{a_k}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$. Then, for all $u \in W$ such that $(w, u) \in R_{a_k}$, either $\mathcal{M}_{[\![\pi]\!]}, u \models \psi$ or $u \in P$, with $P \in \mathcal{P}^{\mathcal{M}}_{\pi}$, and u is at position $k+1$ (because w is at position $k = i$), i.e., $\mathcal{M}, u \models del^{\pi}_{k+1}$ (by Lemma 9.3.6). Therefore, $\mathcal{M}, w \models \Box_{a_k}(del^{\pi}_{k+1} \vee [\![\pi]\!]\psi)$. □

The next theorem establishes that we can reduce $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$-formulas according to axioms of Definition 9.3.7, obtaining an equivalent $\mathcal{ML}(\Diamond^{-1})$-formula.

Theorem 9.3.12. *Let $\mathcal{M} = \langle W, R, V\rangle$ a model, $w \in W$ and φ a $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$-formula. Then*

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}, w \models \mathsf{Tr}(\varphi).$$

Proof. The proof is a direct corollary of Lemmas 9.3.8 to 9.3.10, Theorem 9.3.11, and the similar properties for $\Box^{-1}_{a_i}$. □

We have proved that the reduction axioms from Definition 9.3.7 preserve equivalence of formulas. It means that we can transform $\mathcal{ML}([\![\ ]\!])$ formulas into $\mathcal{ML}(\Diamond^{-1})$-formulas. Then, next Theorem follows:

Theorem 9.3.13. *The satisfiability problem for $\mathcal{ML}([\![\ ]\!])$ is decidable.*

Summing up, in this chapter we discussed a relation-changing modal language which embeds a restricted version of $\mathcal{AML}$. This language includes an operator to delete edges ($[\![\ ]\!]$) and an operator to replicate models (cp). The main difference between $[\![\ ]\!]$ and sabotage operators we investigated previously in this thesis, is that deletions done by $[\![\ ]\!]$ depend of a parameter π (called *path expression*) which characterizes the edges to be removed. Path expressions can describe the label of the edges, properties satisfied by states (only Boolean properties) and composition of edges. Because we do not allow modal characterization of states, the embedding from $\mathcal{AML}$ to $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$ only allows Boolean preconditions in action models.

We investigated some properties of the language $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$. First, we showed that we need the same conditions as for the $\Diamond^{-1}$ operators to give an appropriate notion of bisimulation for the new language. The reason is we need to differentiate states according what paths traverse them, then we need to check conditions back and forth. We prove Theorem 9.1.12, which says that bisimilar models (according to the definition we just mentioned) satisfy the same $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$-formulas. We also investigated the computational behaviour, by inspecting two fragments of the language separately. First, we provided a PSpace algorithm which checks satisfiability for formulas in the fragment $\mathcal{ML}(\mathsf{cp})$ (the language without $[\![\,]\!]$), concluding that the satisfiability problem for $\mathcal{ML}(\mathsf{cp})$ is PSpace-complete. Secondly, we defined an equivalence preserving translation from the fragment $\mathcal{ML}([\![\,]\!])$ to $\mathcal{ML}(\Diamond^{-1})$ via reduction axioms, showing decidability of the satisfiability problem for $\mathcal{ML}([\![\,]\!])$. These reduction axioms possibly generate formulas of exponential size (in the size of the original formula), then we were not able to provide an upper bound yet.

There are several open questions about the language we introduced in this chapter. For instance, we obtained separate results for two fragments of the language $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$, but it would be interesting to know what is the computational behaviour of the combined fragments. Given that we can embed a restricted version of $\mathcal{AML}$ into $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$, we can get next result:

Theorem 9.3.14. *The satisfiability problem for $\mathcal{ML}(cp, [\![\,]\!])$ is* NExpTime*-hard.*

Proof. We embed a restricted version of $\mathcal{AML}$ into $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$, and the satisfiability problem of $\mathcal{AML}$ is NExpTime-hard [Aucher and Schwarzentruber, 2013]. This proof includes the full language (allowing modal preconditions in action models), but the encoding of the tiling problem only uses Boolean formulas in action preconditions. Hence, the satisfiability problem of $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$ is also NExpTime. □

The previous theorem gives us a lower bound for the satisfiability problem of $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$. The upper bound of the problem is still open, even though we conjecture it is decidable.

Another interesting direction of research would be to define concrete procedures to check satisfiability (e.g., tableaux), and to investigate other reasoning tasks, such as the model checking problem, formula complexity and program complexity. In addition, we have not yet discussed the relationship between $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$ and other relation-changing operators, such as those introduced in the first part of this thesis (sabotage, swap and bridge) and others (arrow updates, graph modifiers, etc.). Finally, it would be interesting to extend the definition of path expressions to allow nested deletions and modal formulas, to embed the full $\mathcal{AML}$ language with modal preconditions. We would like to explore the computational behaviour of this extended language in the future.

Part IV

CONCLUSIONS

Un cronopio encuentra una flor solitaria en medio de los campos. Primero la va a arrancar, pero piensa que es una crueldad inútil y se pone de rodillas a su lado y juega alegremente con la flor a saber: le acaricia los pétalos, la sopla para que baile, zumba como una abeja, huele su perfume, y finalmente se acuesta debajo de la flor y se duerme envuelto en una gran paz. La flor piensa: 'es como una flor'.

from "Historias de Cronopios y de Famas", Julio Cortázar.

Now is when the end of the thesis starts and, in consequence, it is when new questions arise. Time for finishing a thesis is limited, but the work we have done opens many paths of research. It is time to make a balance of what we have learned so far, and what it would be interesting to explore in the future.

We have introduced several modal languages to represent dynamic behaviour. We focused on a particular kind of logics which let us change the accessibility relation. We have applied some classical techniques from modal logics to prove these results, such as spy point techniques or proofs based on bisimulation. We explored different properties of these languages: expressive power, computational behaviour of reasoning tasks, and the existence of algorithms to perform these reasoning tasks. We have observed that the high expressive power we obtained by using dynamic operators has a vast impact on the cost of reasoning about these languages. The connection we established between the languages we introduced and other languages investigated in other fields, such as epistemic, doxastic or deontic logics is important. It gives us a clear direction to follow in the future.

We still have work to do, and we will discuss new lines of investigation. The challenge is to exploit the experience we gained with this kind of languages in different contexts. In Chapter 10 we will brush up what we have done and what we foresee for the future.

10 Final Remarks

El porvenir es tan irrevocable como el rígido ayer.
No hay una cosa que no sea una letra silenciosa
de la eterna escritura indescifrable cuyo libro es el tiempo.
Quien se aleja de su casa ya ha vuelto.
Nuestra vida es la senda futura y recorrida.
Nada nos dice adiós. Nada nos deja. No te rindas.
La ergástula es oscura, la firme trama es de incesante hierro,
pero en algún recodo de tu encierro puede haber un descuido, una hendidura.
El camino es fatal como la flecha. Pero en las grietas está Dios, que acecha.

from "Para una versión del I King", Jorge Luis Borges.

We started this thesis by discussing the criteria to be used when picking a logic in a determined scenario. In Part I we pointed to one particular question: *"Why do we choose dynamic modal logics?"*. Before going in this direction, we briefly discussed why logic was born in philosophy, became well known in mathematics and now, is extremely tied to computer science. In Chapter 1 we discussed several examples of the use of logic to reason about computation, i.e., *Computational Logic*. One example is the use of logic in complexity theory to give descriptive characterizations of the complexity classes (e.g., NTime, NP, PSpace, ExpTime, etc.). Another example is logic applied in databases theory. Several logical languages can be seen as Database Query Languages, such as SQL (Structured Query Language) or QBE (Query by Example), which are based on first-order logic. Also, we mentioned that logic can be used to reason about program behaviour in type theory, and to reason about knowledge and belief in epistemic logics. Finally, we discussed applications of logic in software verification, perhaps the most common use of logic in computer science. As we can see, depending of the problem we need to solve, we can pick the logic that is better suited for that. This is the reason why we talk about *logics* (plural) instead of *logic*. We introduced first-order logic just to emphasize why other logics can be more appropriate than first-order for our purposes. If we define our own languages we are able to adapt them to our needs. For instance, we can restrict ourselves to fragments with the desired properties.

In Chapter 2, we started to discuss the main problem we attacked in this thesis: representing dynamic behaviour. This is the chapter which motivated our work, by introducing several languages that capture dynamic behaviour. These languages are ancestors of the operators that we introduced in Part II. We discussed various approaches and we motivated why we are particularly interested in this kind of operators. We noticed that even though dynamic operators have been investigated already in the past, in this thesis we would focus on the behaviour of those which modify the accessibility relation, from an abstract point of view. Our main purpose is to analyze the behaviour of certain kind of modifications, and the advantages and disadvantages of the use of these languages.

10.1 WHAT HAVE WE DONE?

Part II is dedicated to analyzing relation-changing modal logics from an abstract point of view. We investigated different logical aspects of these languages, mostly from a computational perspective. We chose a set of primitives as representatives of the family of all possible relation-changing operators and we studied in detail each of them separately. We also studied the relationships among them. The main contributions of our work have been, on the one hand, particular results for the six logics we introduced and, on the other hand, general results that can be applied to a bigger family of logics. We learned much about the behaviour of these six relation-changing modal logics, and we can use this experience to establish more general results.

In Chapter 3 we introduced the formal syntax and semantics of six relation-changing modal logics: sabotage, swap and bridge, each of them in a local and global version. Each language is a syntactic extension of the basic modal logic $\mathcal{ML}$ with a dynamic operator. Semantics is based on Kripke models and *model variants*, which are operations which modify the model capturing the behaviour of the new syntactic operators. We introduced "ad hoc" model variants for each operator, but clearly it would be easy to define modal variants based on generic update functions. We observe that, even though the semantics conditions look innocent, the operators capture complex behaviour resulting in a high increase in expressive power. Indeed, we showed that two classical model theoretical properties for modal logics are lost: the tree and the finite model property. For the tree model properties, in most cases we defined formulas that enforce loops, and for bridge operators we enforced unconnected components. Formulas to force infinite models are more complex. In order to achieve that, we use a classical tool to prove expressiveness results in logic: *the spy point technique*. The idea is to characterize a state which has a global view of the rest of the model (the "spy point"), and use it to describe the properties we want to impose in the model.

We showed that the relation-changing modal logics we introduced are more expressive that $\mathcal{ML}$ by proving they lack the tree and the finite model property. In Chapter 4 we obtained more expressivity results. The main tool used in modal logic to investigate expressive power is *bisimulation*. Bisimulations are relations that link states of models according to their behaviour. If two states in two models are linked by a bisimulation, then they must satisfy the same formulas. This is called *invariance under bisimulation*. In [Areces *et al.*, 2012; Areces *et al.*, 2013b] we introduced an appropriate definition of bisimulation for each operator, capturing their meaning. We proved in [Areces *et al.*, 2013b] that $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ is a proper fragment of $\mathcal{FOL}$ and we conjecture that the same arguments can be applied for the other five logics. However, they all capture different fragments: they are all incomparable in expressive power except the case for the two swap operators. We know that $\mathcal{ML}(\langle \mathsf{gsw} \rangle) \not\leq \mathcal{ML}(\langle \mathsf{sw} \rangle)$, but the other direction is still open. Nevertheless, we conjecture that they are also incomparable.

Another challenge has been to establish computational bounds. We explore in Chapter 5 the satisfiability problem of these logics. We attacked the problem in two different ways: by encoding the undecidable $\mathbb{N}\times\mathbb{N}$ tiling problem, and by encoding the undecidable satisfiability problem for the memory logic $\mathcal{ML}(ⓡ, ⓚ)$. The second approach gave better results, and we applied it to the local version of the logics. In order to encode $\mathcal{ML}(ⓡ, ⓚ)$, we use once more a spy point technique to simulate the capability to memorize elements without a memory. This is one of the hardest results in this thesis: it required to enforce some constraints in the shape of the models, and a different machinery to simulate memory operators with each of the relation-changing operators. We conjecture that similar proofs can be carried out for the global versions of the relation-changing operators.

The satisfiability problem is not the only reasoning task we have investigated. In Chapter 6 we proved complexity results for the model checking problem. We reduced the PSpace-complete satisfiability problem for Quantified Boolean Formulas (QBF) to the model checking problem of relation-changing modal logics. This proves PSpace-hardness for the six model checking problems. The encoding is in general, fairly uniform, requiring minor modifications to adapt it to each different operator. We simulate truth assignments to QBF variables with the position of the edges in a Kripke model, and changes in their truth values are simulated by changes in the accessibility relation. We also considered the task of model checking against a fixed model, measuring its complexity as a function of the size of an input formula (formula complexity), and fixing a formula and measuring the complexity of model checking as a function of the length of an input model (program or data complexity). We proved that formula complexity for the six logics is linear with respect to the size of the formula, and program complexity is polynomial with respect to the size of the finite model. Interestingly, we were able to prove these results for a large family of relation-changing logics. We defined relation-changing operators in terms of update functions (i.e., functions that transform the accessibility relation). For any logic, if we have that the update functions associated with its operators result in only polynomially many possible models in each step, our formula and program complexity results apply.

Clearly, including relation-changing operators (at least the six we investigated in this thesis) results in some reasoning tasks turning intractable. However, it is possible to define concrete procedures whose termination is not guaranteed. In Chapter 7 we defined tableau methods to check satisfiability of relation-changing modal logics. This work was first published in [Areces *et al.*, 2013c]. These methods are complete and sound, but they may not terminate. Tableaux formulas in the calculi we defined, contain prefixes, which are quite different to prefixes in other tableau methods (e.g., for $\mathcal{ML}$, $\mathcal{HL}(@, \downarrow)$, etc.). In our tableaux, prefixes provide the information about the evaluation point and also the model variant where the formula is been executed. For this reason, we had to defined different rules for each logic (except for the local an global version of the same modifier). For instance, sabotage prefixes keep a record of the deleted edges. As it has been the case in several results, swap operators have been the hardest to be investigated. We have to consider some special cases such as when the swapped edge is a loop (do not provoke change at all) or when the swapped edge has been swapped before. The rules for each logic are very different, and it seems hard to find an uniform framework that would allow combination to define a tableau calculi for a logic with more than one relation-changing operator (we can only combine the local and the global version of the same operator).

Our results gave us experience investigating operators which can modify the accessibility relation of a model. Other operators which fall in this class but which have been designed for a determined purpose have been investigated in previous works. One example are the operators of information change in *Dynamic Epistemic Logics* ($\mathcal{DEL}$). It resulted natural to investigate connections between these operators and the operators we defined. The first question which arose was: *"Is it possible to encode dynamic epistemic logics with some variant of the six logics we introduced?"*. Chapter 8 is devoted to introduce $\mathcal{DEL}$ and to discuss why relation-changing is important in this field. In $\mathcal{DEL}$, models represent information and knowledge of agents, and dynamic operators are used to model communication, i.e., the way in which agents' knowledge is modified. We define a new relation-changing language to work in this field. The language we defined in Chapter 9 is more expressive than $\mathcal{ML}$, but its computational behaviour is better than the one for the six relation-changing modal logics we investigated before.

10.2 LOOKING TO THE FUTURE

As we already pointed out in the different chapters, several questions are still open. This thesis addressed many questions about relation-changing modal logics. But of course time is limited and there are some interesting directions to follow in the future.

We conjecture, for example, that $\mathcal{ML}(\langle\mathsf{sw}\rangle)$ and $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$ are incomparable in expressive power, and that the logic $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ is decidable. Also, we mentioned that the proof of undecidability of the satisfiability problem for the local operators can be adapted for the global. It would be interesting to finish these results to have a complete picture of the results introduced in this thesis.

Concerning expressive power, we would like to define a more general notion of bisimulation which instead of considering particular cases of updates (deleting, swapping or adding edges) might consider update functions in general (similar to the ones we used for our model checking results). The same ideas can also be applied, for instance, to Dynamic Epistemic Logics. For $\mathcal{PAL}$ and $\mathcal{AML}$, it suffices with the notion of bisimulation of the basic modal logic $\mathcal{ML}$. If we extend the language with more expressive operators, it would be possible to introduce update conditions in the same way we did for relation-changing modal logics. We also mentioned in Chapter 4 that it would be interesting to investigate *van Benthem Characterization* for relation-changing modal logics. We can start by checking if results introduced in [Areces *et al.*, 2013a] can be used for relation-changing modal logics. Another interesting property we started to investigate is Craig's Interpolation Lemma. We studied the constructive method provided in [Fitting, 1996; Fitting, 2002; Blackburn and Marx, 2003] to follow the same ideas in relation-changing modal logics, but this is left as future work.

Bibliography

[Alur and Henzinger, 1994] R. Alur and T. Henzinger. A really temporal logic. *Journal of ACM*, 41(1):181–204, 1994. Cited on page 20.

[Areces and Gorín, 2010] C. Areces and D. Gorín. Coinductive models and normal forms for modal logics (or how we learned to stop worrying and love coinduction). *Journal of Applied Logic*, 8(4):305–318, 2010. Cited on page 70.

[Areces and ten Cate, 2006] 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. Cited on page 15.

[Areces *et al.*, 1999] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in Lecture Notes in Computer Science, pages 307–321, Madrid, Spain, 1999. Springer. Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 1999. Cited on page 15.

[Areces *et al.*, 2001] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: characterization, interpolation and complexity. *The Journal of Symbolic Logic*, 66(3):977–1010, 2001. Cited on page 15.

[Areces *et al.*, 2008] C. Areces, D. Figueira, S. Figueira, and S. Mera. Expressive power and decidability for memory logics. In *Logic, Language, Information and Computation*, volume 5110 of *Lecture Notes in Computer Science*, pages 56–68. Springer Berlin / Heidelberg, 2008. Proceedings of WoLLIC 2008. Cited on page 17.

[Areces *et al.*, 2009] C. Areces, D. Figueira, D. Gorín, 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. Cited on pages 17 and 88.

[Areces *et al.*, 2011] C. Areces, D. Figueira, S. Figueira, and S. Mera. The expressive power of memory logics. *The Review of Symbolic Logic*, 4(2):290–318, 2011. Cited on page 16.

[Areces *et al.*, 2012] C. Areces, R. Fervari, and G. Hoffmann. Moving arrows and four model checking results. In L. Ong and R. Queiroz, editors, *Logic, Language, Information and Computation*, volume 7456 of *Lecture Notes in Computer Science*, pages 142–153. Springer Berlin Heidelberg, 2012. Cited on pages 21, 33, 43, 48, 65, 71, and 130.

[Areces *et al.*, 2013a] C. Areces, F. Carreiro, and S. Figueira. Characterization, definability and separation via saturated models. *Theoretical Computer Science*, 2013. Cited on pages 51 and 132.

[Areces *et al.*, 2013b] C. Areces, R. Fervari, and G. Hoffmann. Swap logic. *Logic Journal of IGPL*, 2013. Cited on pages 21, 33, 43, 53, 65, 71, and 130.

[Areces *et al.*, 2013c] C. Areces, R. Fervari, and G. Hoffmann. Tableaux for relation-changing modal logics. In P. Fontaine, C. Ringeissen, and R. Schmidt, editors, *Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 263–278. Springer, 2013. Cited on pages 21, 71, 75, and 131.

[Aucher and Schwarzentruber, 2013] G. Aucher and F. Schwarzentruber. On the complexity of dynamic epistemic logic. In *Proceedings of TARK 2013*, Chennai, India, January 2013. Cited on pages 106 and 124.

[Aucher *et al.*, 2009] G. Aucher, P. Balbiani, L. Fariñas Del Cerro, and A. Herzig. Global and local graph modifiers. *Electronic Notes in Theoretical Computer Science (ENTCS), Special issue Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007)*, 231:293–307, 2009. Cited on pages 19, 25, 29, and 95.

[Baltag *et al.*, 1998] A. Baltag, L. Moss, and S. Solecki. The logic of public announcements, common knowledge and private suspicions. In I. Gilboa, editor, *TARK 1998*, pages 43–56, Evanstin, IL, USA, July 1998. Morgan Kaufmann. Cited on page 102.

[Berger, 1966] R. Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66:72, 1966. Cited on page 8.

[Beth, 1955] E. W. Beth. Semantic entailment and formal derivability. *Mededellingen van de Koninklijke Nederlandse Akademie van Wetenschappen Afdeling Letterkunde N. R.*, 18(13):309–342, 1955. Cited on page 75.

[Blackburn and Marx, 2003] P. Blackburn and M. Marx. Constructive interpolation in hybrid logic. *Journal of Symbolic Logic*, 68(2):463–480, 2003. Cited on pages 89 and 132.

[Blackburn and Seligman, 1995] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995. Cited on page 15.

[Blackburn and van Benthem, 2006] P. Blackburn and J. van Benthem. Modal logic: A semantic perspective. In *Handbook of Modal Logic*. Elsevier North-Holland, 2006. Cited on pages 9, 11, 32, and 114.

[Blackburn *et al.*, 2001] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2001. Cited on pages 9, 35, 112, and 114.

[Chandra and Merlin, 1977] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of 9th ACM Symposium on Theory of Computing*, pages 77–90, 1977. Cited on page 9.

[Church, 1936] A. Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1:40–41, 1936. Cited on page 8.

[Clarke, 2008] E. M. Clarke. The birth of model checking. In O. Grumberg and H. Veith, editors, *25 Years of Model Checking*, volume 5000 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2008. Cited on page 6.

[Craig, 1957] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22:269–285, 1957. Cited on page 89.

[D'Agostino *et al.*, 1999] M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga. *Handbook of tableau methods*. Kluwer Academic Publishers, 1999. Cited on page 75.

[Demri *et al.*, 2007] S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in constraint LTL: Decidability and complexity. *Information and Computation*, 205(1):2–24, January 2007. Cited on page 20.

[Dummet and Lemmon, 1959] M. Dummet and E. Lemmon. Modal logics between s4 and s5. *Zeitschrift für mathemathische Logik und Grundla-gen der Mathematik*, 5:250–264, 1959. Cited on page 32.

[Ebbinghaus *et al.*, 1984] H. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, 1984. Cited on page 42.

[Enderton, 1972] H. Enderton. *A mathematical introduction to logic*. Academic Press, 1972. Cited on page 7.

[Fagin, 1974] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *SIAM-AMS Proceedings*, volume 7, pages 43–73, 1974. Cited on page 6.

[Fervari, 2012] R. Fervari. The expressive power of swap logic. In *Proceedings of ESSLLI Student Session*, Opole, Poland, 2012. Cited on page 21.

[Fischer and Ladner, 1979] M. Fischer and R. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979. Cited on page 12.

[Fitting, 1972] M. Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, 13(2):237–247, 1972. Cited on page 75.

[Fitting, 1996] M. Fitting. *First-order logic and automated reasoning (2. ed.)*. Graduate texts in computer science. Springer, 1996. Cited on pages 89 and 132.

[Fitting, 2002] M. Fitting. Interpolation for first order s5. *Journal of Symbolic Logic*, 67(2):621–634, 2002. Cited on pages 89 and 132.

[Franceschet and de Rijke, 2003] M. Franceschet and M. de Rijke. Model checking for hybrid logics. In *Proceedings of the 3rd International Workshop on Methods for Modalities (M4M*, pages 109–123, 2003. Cited on page 15.

[French *et al.*, 2013] T. French, W. van der Hoek, P. Iliev, and B. Kooi. On the succinctness of some modal logics. *Artificial Intelligence*, 197:56–85, 2013. Cited on page 106.

[Goranko and Otto, 2005] 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. Cited on page 43.

[Goré, 1999] R. Goré. Tableau methods for modal and temporal logics. *Handbook of tableau methods*, pages 297–396, 1999. Cited on page 118.

[Grädel *et al.*, 1997] E. Grädel, P. Kolaitis, and M. Vardi. On the decision problem for two-variable first-order logic. *The Bulletin of Symbolic Logic*, 3(1):53–69, 1997. Cited on page 10.

[Groenendijk and Stokhof, 1991] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14(1):39–100, 1991. Cited on page 19.

[Halpern *et al.*, 2001] J. Y. Halpern, R. Harper, N. Immerman, P. G. Kolaitis, M. Y. Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *Bulletin of Symbolic Logic*, 7(2):213–236, 2001. Cited on page 5.

[Harel, 1984] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic. Vol. II*, volume 165 of *Synthese Library*, pages 497–604. D. Reidel Publishing Co., Dordrecht, 1984. Extensions of classical logic. Cited on page 12.

[Hemaspaandra, 1996] E. Hemaspaandra. The price of universality. *Notre Dame Journal of Formal Logic*, 37(2):174–203, 1996. Cited on page 12.

[Hintikka, 1962] J. Hintikka. *Knowledge and Belief. An Introduction to the Logic of the Two Notions.* Cornell University Press, Ithaca, NY, 1962. Cited on pages 6 and 96.

[Immerman, 1995] N. Immerman. Descriptive complexity: A logician's approach to computation. *Notices of the American Mathematical Society*, 42, 1995. Cited on page 6.

[Kooi and Renne, 2011a] B. Kooi and B. Renne. Arrow update logic. *Review of Symbolic Logic*, 4(4):536–559, 2011. Cited on pages 17, 18, 25, 29, and 95.

[Kooi and Renne, 2011b] B. Kooi and Bryan Renne. Generalized arrow update logic. In K. Apt, editor, *TARK*, pages 205–211. ACM, 2011. Cited on page 25.

[Kripke, 1963] S. Kripke. Semantical analysis of modal logic I. Normal propositional calculi. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963. Cited on page 96.

[Kupferman *et al.*, 2000] O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000. Cited on page 70.

[Ladner, 1977] R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977. Cited on page 12.

[Lewis, 1918] C. Lewis. *A Survey of Symbolic Logic*. University of California Press, 1918. Republished by Dover, 1960. Cited on page 13.

[Löding and Rohde, 2003a] C. Löding and P. Rohde. Model checking and satisfiability for sabotage modal logic. In P. Pandya and J. Radhakrishnan, editors, *Proceedings of Foundations of Software Technology and Theoretical Computer Science, 23rd Conference*, volume 2914 of *Lecture Notes in Computer Science*, pages 302–313. Springer, 2003. Cited on pages 19, 33, 65, and 70.

[Löding and Rohde, 2003b] C. Löding and P. Rohde. Solving the sabotage game is PSPACE-hard. In *Mathematical Foundations of Computer Science 2003*, volume 2747 of *Lecture Notes in Computer Science*, pages 531–540. Springer, Berlin, 2003. Cited on pages 19, 25, and 65.

[Lutz, 2006] C. Lutz. Complexity and succinctness of public announcement logic. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *AAMAS*, pages 137–143, Hakodate, Japan, 2006. ACM. Cited on page 106.

[Maier, 1983] D. Maier. *The theory of relational databases*. Number v. 1 in Computer software engineering series. Computer Science Press, 1983. Cited on page 6.

[Mera, 2009] S. Mera. *Modal Memory Logics*. PhD thesis, Universidad de Buenos Aires Facultad de Ciencias Exactas y Naturales Departamento de Computación, and UFR STMIA - Ecole Doctorale IAEM Lorraine Département de Formation Doctorale en Informatique, 2009. Cited on pages 16 and 17.

[Papadimitriou, 1994] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. Cited on pages 65 and 66.

[Plaza, 2007] J. Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007. Cited on pages 12, 18, and 101.

[Prior, 1957] A. Prior. *Time and Modality*. Oxford University Press, 1957. Cited on page 13.

[Pucella and Weissman, 2004] R. Pucella and V. Weissman. Reasoning about dynamic policies. In I. Walukiewicz, editor, *FoSSaCS*, volume 2987 of *Lecture Notes in Computer Science*, pages 453–467. Springer, 2004. Cited on pages 19 and 93.

[Reynolds, 1985] J. C. Reynolds. Three approaches to type structure. In H. Ehrig, C. Floyd, M. Nivat, and J. W. Thatcher, editors, *TAPSOFT, Vol.1*, volume 185 of *Lecture Notes in Computer Science*, pages 97–138. Springer, 1985. Cited on page 6.

[Reynolds, 1998] J. C. Reynolds. *Theories of programming languages*. Cambridge University Press, 1998. Cited on page 6.

[Rohde, 2006] P. Rohde. *On games and logics over dynamically changing structures*. PhD thesis, RWTH Aachen, 2006. Cited on pages 19, 25, 30, 70, and 71.

[Sahlqvist, 1973] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logic. In S. Kanger, editor, *Third Scandinavian Logic Symposium*, pages 110–143, Uppsala, 1973. North-Holland Publishing Company 1975. Cited on page 32.

[Sangiorgi, 2009] D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(4), 2009. Cited on page 42.

[Schmidt and Tishkovsky, 2007] R. A. Schmidt and D. Tishkovsky. Using tableau to decide expressive description logics with role negation. In *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 438–451. Springer, 2007. Cited on page 77.

[Schnoebelen, 2002] P. Schnoebelen. The complexity of temporal logic model checking. In P. Balbiani, N. Suzuki, F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logic 4*, pages 393–436. King's College Publications, 2002. Cited on page 70.

[Scott, 1962] D. Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:377, 1962. Cited on page 10.

[Smullyan, 1968] R. Smullyan. *First-Order Logic*. Springer-Verlag, 1968. Cited on page 75.

[Spaan, 1993] E. Spaan. *Complexity of Modal Logics*. PhD thesis, ILLC, University of Amsterdam, 1993. Cited on page 12.

[Stockmeyer, 1974] L. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, MIT, Cambridge, Mass., 1974. Cited on page 9.

[ten Cate, 2005] B. ten Cate. *Model theory for extended modal languages*. PhD thesis, ILLC, University of Amsterdam, 2005. ILLC Dissertation Series DS-2005-01. Cited on page 15.

[Turing, 1937] A. Turing. On computable numbers, with an application to the 'Entscheidungsproblem'. *Proceedings of the London Mathematical Society 2nd. series*, 42:230–265, 1937. Cited on page 8.

[van Benthem and Liu, 2007] J. van Benthem and F. Liu. Dynamic logic of preference upgrade. *Journal of Applied Non-Classical Logics*, 17(2):157–182, 2007. Cited on page 102.

[van Benthem, 1977] J. van Benthem. *Modal Correspondence Theory*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1977. Cited on page 51.

[van Benthem, 1984] J. van Benthem. Modal correspondence theory. *Handbook of Philosophical Logic*, 2:167–247, 1984. Cited on page 41.

[van Benthem, 1985] J. van Benthem. *Modal logic and classical logic*. Bibliopolis, 1985. Cited on page 41.

[van Benthem, 2001] J. van Benthem. Games in dynamic-epistemic logic. *Bulletin of Economic Research*, 53(4):219–48, 2001. Cited on page 104.

[van Benthem, 2005] J. van Benthem. An essay on sabotage and obstruction. In D. Hutter and W. Stephan, editors, *Mechanizing Mathematical Reasoning*, volume 2605 of *Lecture Notes in Computer Science*, pages 268–276. Springer, 2005. Cited on pages 18, 25, and 27.

[van Ditmarsch *et al.*, 2007] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Kluwer, 2007. Cited on pages 12, 18, 29, 70, 93, 95, and 96.

[Vardi and Wolper, 1986] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344. IEEE Computer Society, 1986. Cited on page 70.

[Vardi, 1982] M. Vardi. The complexity of relational query languages. In H. Lewis, B. Simons, W. Burkhard, and L. Landweber, editors, *Symposium on Theory of Computing*, pages 137–146. ACM, 1982. Cited on pages 9 and 70.

[von Wright, 1951] G. H. von Wright. *An Essay in Modal Logic*. Amsterdam, North-Holland Pub. Co., 1951. Cited on pages 6, 13, and 96.

Index