Universidad Nacional de Río Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Departamento de Computación

On the Undecidability of Relation-Changing Logics

Thesis presented to obtain the degree of
Licenciado en Ciencias de la Computación

Mauricio Martel

Supervisor
Dr. Carlos Areces

Co-Supervisor
Dr. Pablo Castro

March, 2015

Contents

Acknowledgments

My studies in Computer Science started a long time ago, but it was only just a few years back when I first heard about logics that could change the model, and then, the adventure began. I have been supported and helped by several people throughout this thesis, and I would like to express my gratitude to them.

My greatest thanks goes to my supervisor, Carlos Areces. Thanks for giving me the opportunity to work on the topic of this thesis and for introducing me to the fascinating world of Modal Logic. His support and flexibility made it a pleasure to have him as a supervisor. Thanks also for letting me be part of the Logics, Interaction and Intelligent Systems Group (LIIS), which turned working on this thesis into the most valuable experience I had during my studies.

I also want to express my gratitude to the people who worked with me in the results of this thesis: Raul Fervari and Guillaume Hoffmann. Thanks for guiding me and helping me understand how these relation-changing logics behave. Having worked on this thesis would not have been so pleasant and enjoyable without their support. I am also grateful to Pablo Castro for accepting to be my co-supervisor and for taking the time to read earlier drafts of this thesis.

Thanks to all the people of the Department of Computer Science in Río Cuarto. The Department has excellent professors teaching us, and because of that, it was possible for me to organize crazy things such as talks and trips to conferences. Thanks to all of them. In the same way, I am grateful to all the people of the Department of Mathematics in Río Cuarto. In particular, a special thanks goes to María Elena Markiewicz, who awakened my interests in Logic and Algebra. Thanks for her encouragement, for all the meetings we had, and for the many enriching discussions.

Finally, I would like to thank my family for their support and for always encouraging me in every decision I take.

Mauricio Martel
Río Cuarto, Argentina
March, 2015

Abstract

We can fairly say that Logic (whichever you want to choose, be it propositional or first order, classical or non-classical) is the mathematical tool used, par excellence, to describe a structure. Modal logics, for example, are particularly well suited to describe relational structures, especially if you are interested in computationally well behaved formalisms (the model checking problem of the basic modal logic is only polynomial, while its satisfiability problem is decidable and PSPACE-complete). But why can a logic only *describe* a structure? In this thesis we introduce a family of modal logics that contain operators which can both describe and *change* the structure. We are interested in modal operators that allow to change the relational model, i.e, that allow to change the structure of a graph. In particular, we extend the basic modal language with modalities that are able to delete, add, or swap pairs of related elements of the domain. These *dynamic* operators can work locally (changing adjacent edges from the evaluation point) or globally (modifying edges anywhere in the model). It has been shown that the resulting logics have greater expressive power than the basic modal language, and the complexity of their model checking problems has been proven to be PSPACE-complete. In this thesis we address the question of decidability. We prove that the satisfiability problems of the logics we introduce are all undecidable.

CHAPTER 1

Introduction

Modal logic was originally conceived as the logic of necessary and possible truths. It is now view more broadly as the logic of different sorts of modalities, or modes of truth: epistemic ("it is known that"), doxastic ("it is believed that"), deontic ("it ought to be the case that"), or temporal ("it has been the case that") among others. Moreover, modal logic has developed into a powerful mathematical discipline that deals with languages for talking about various kinds of relational structures.

We will start this chapter giving a brief historical overview of modal logic [Gol06] in order to understand what have been done in the past, and at the same time to provide some insight into how the present came to be as it is. Hopefully, this introduction will give the reader some idea of what this thesis is going to talk about and what new contribution to knowledge this thesis is going to make. Let us begin.

1.1 Modern Origins of Modal Logic

The origins of modal logic as a mathematical discipline can be traced back at the turn of the twentieth century, with the work of C. I. Lewis, who tried to solve the paradoxes of material implication using necessity and possibility. In fact there were other people before Lewis who built logic systems for this purpose, but Lewis seems to be the strongest link with contemporary modal logic.

The Lewis Systems

In a 1912 pioneering article, "Implication and the Algebra of Logic" [Lew12], Lewis started to voice his concerns on the so-called "paradoxes of material implication." Lewis observed that the algebraic meaning of implication as used in Russell and Whitehead's Principia Mathematica, leads to two "startling theorems": (1) a false proposition implies any proposition, and (2) a true proposition is implied by any proposition. In symbols:

$$(1)\ \neg\alpha \to (\alpha \to \beta)$$
$$(2)\quad \alpha \to (\beta \to \alpha)$$

For Lewis the ordinary meaning of $\alpha \to \beta$ is that β can be inferred from α, an interpretation that he considered was not subject to these paradoxes. Taking $\alpha \to \beta$

as synonymous with $\neg\alpha \vee \beta$, he distinguished *extensional* and *intensional* meanings of disjunction, providing two meanings for implication. On the one hand, extensional disjunction is the usual truth-functional disjunction, meaning that "it is false that α is true and β is false." That reading gives meaning to the *material* (*algebraic*) implication of Principia Mathematica. On the other hand, intensional disjunction is such that at least one of the disjoined propositions is *necessarily* true, meaning that "it is *impossible* that α is true and β is false." That reading gives meaning to the *ordinary* (*strict*) implication of Lewis.

In his 1918 book, "A Survey of Symbolic Logic" [Lew18], Lewis introduces a first system meant to capture the ordinary, strict sense of implication. The 1918 system used as primitive an *impossibility* operator to define strict implication. Then, in Appendix II of the Lewis and Langford's volume "Symbolic Logic" [LL32], the 1918 system is given a new axiomatic base. In the 1932 Appendix Lewis introduces five different systems, S1-S5. The systems were defined with negation, conjunction, and possibility ($\Diamond$) as their primitive connectives. Intuitively, $\Diamond\alpha$ asserts that α is *possibly* true. Dually, $\neg\Diamond\neg\alpha$, abbreviated as $\Box\alpha$, asserts that α is *necessary* true, though Lewis made no use of the symbol $\Box$ for the dual combination. For strict implication we use the symbol $\Rightarrow$, with $\alpha \Rightarrow \beta$ being a definitional abbreviation for $\neg\Diamond(\alpha \wedge \neg\beta)$. Strict equivalence ($\alpha = \beta$) is defined as $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

Here are now definitions of S1-S5. Where p, q, r are propositional variables, system S1 has the following axioms[1]:

$$
\begin{aligned}
&(p \wedge q) \Rightarrow (q \wedge p)\\
&(p \wedge q) \Rightarrow p\\
&p \Rightarrow (p \wedge p)\\
&((p \wedge q) \wedge r) \Rightarrow (p \wedge (q \wedge r))\\
&((p \Rightarrow q) \wedge (q \Rightarrow r)) \Rightarrow (p \Rightarrow r)\\
&(p \wedge (p \Rightarrow q)) \Rightarrow q,
\end{aligned}
$$

and the following rules of inference:

- *Uniform substitution* of formulas for propositional variables.
- *Substitution of strict equivalents*: from $(\alpha = \beta)$ and γ infer any formula obtained from γ by substituting β for some occurrence(s) of α.
- *Adjunction*: from α and β infer $\alpha \wedge \beta$.
- *Strict detachment*: from α and $\alpha \Rightarrow \beta$ infer β.

System S2 is obtained by adding the axiom $\Diamond(p \wedge q) \Rightarrow \Diamond p$ to the basis for S1. S3 is S1 plus the axiom $(p \Rightarrow q) \Rightarrow (\neg\Diamond q \Rightarrow \neg\Diamond p)$. S4 is S1 plus $\Diamond\Diamond p \Rightarrow \Diamond p$, or equivalently $\Box p \Rightarrow \Box\Box p$. S5 is S1 plus $\Diamond p \Rightarrow \Box\Diamond p$. In general, the Lewis systems are numbered in order of strength, with S1 the weakest and S5 the strongest, weaker systems being contained in the stronger ones.

Alternative Axiomatizations and Other Systems

Gödel in "An Interpretation of the Intuitionistic Propositional Calculus" [Göd33] is the first to propose an alternative axiomatization of the Lewis system S4 that separates the propositional basis of the system from the modal axioms and rules.

[1] Originally $p \Rightarrow \neg\neg p$ was included as an axiom, but was proved redundant by McKinsey in 1934.

Gödel adds the following axioms and rules of inference to the propositional calculus. Axiom K: $\Box(p \to q) \to (\Box p \to \Box q)$, Axiom T: $\Box p \to p$, Axiom 4: $\Box p \to \Box\Box p$, and the rule of *Necessitation*: from α infer $\Box\alpha$. Initially, Gödel employs an operator B of provability, reading $B\alpha$ as "α is provable," to translate Heyting's primitive intuitionistic connectives, and then observes that if we replace B[2] with an operator of necessity we obtain system S4. Gödel also claims that a formula $\Box p \vee \Box q$ is not provable in S4 unless either $\Box p$ or $\Box q$ is provable, analogously to intuitionistic disjunction. Gödel's claim will be proved algebraically by McKinsey and Tarski [MT48]. The translations in Gödel's short note provided an important connection between intuitionistic and modal logic that contributed to the development both of topological interpretations and of Kripke semantics for intuitionistic logic. Its ideas also formed the precursor to the substantial branch of modal logic concerned with the modality "it is provable in Peano arithmetic that" [Boo95].

It is now standard practice to present modal logics in the axiomatic style of Gödel. The notion of a logic refers to any set Λ of formulas that includes all truth-functional tautologies and is closed under the rules of uniform substitution for variables and detachment for material implication. The formulas belonging to Λ are the Λ-theorems, and are also said to be Λ-provable. A logic is called *normal* if it includes axiom K and has the rule of Necessitation. System S5 can be defined as the normal logic obtained by adding the axiom $p \to \Box\Diamond p$ to Gödel's axiomatisation of S4. The axiom $p \to \Box\Diamond p$ is called the Brouwerian axiom. The smallest normal logic is commonly called K, in honour of Kripke. The normal logic obtained by adding the axiom $\Box p \to p$ to K is known as T. That system was first defined by Feys in 1937 by subtracting axiom 4 from Gödel's system S4. In "An Essay in Modal Logic" [vW51] von Wright discusses alethic, epistemic, and deontic modalities, and introduces system M, which is equivalent to Feys' system T. The system B is the normal logic obtained by adding the Brouwerian axiom to T. Systems K, T, S4, and S5 form a nested hierarchy of systems, making up the core of normal modal logic.

Modal Algebras

By the time that the Lewis systems appeared, algebra was well-established as a postulational science, and the study of the very notion of an abstract algebra was being pursued [Bir33, Bir35]. Over the next few years, algebraic techniques were applied to the study of modal systems using *modal algebras*: Boolean algebras with an additional operation to interpret $\Diamond$.

J.C.C. McKinsey [McK41] showed that there is an algorithm for deciding whether any given formula is a theorem of S2, and likewise for S4. His method was to show that if a formula is not a theorem of the logic, then it is falsified by some finite model which satisfies the logic. This property was dubbed the *finite model property* by Ronald Harrop [Har58], who proved the general result that any finitely axiomatisable propositional logic Λ with the finite model property is decidable.

McKinsey used models of the form $(K, D, -, ^*, \cdot)$, called matrices, where $-$, *, $\cdot$ are operations on a set K for evaluating the connectives $\neg$, $\Diamond$, and $\wedge$, while D is a set of designated elements of K. A formula α is *satisfied* by such a matrix if every assignment of elements of K to the variables of α results in α being evaluated to a member of the subset D. A logic is *characterised* by a matrix if the matrix satisfies the theorems of the logic and no other formulas. And a matrix is *normal* if

[2]The name B comes from "beweisbar," which is the German word for "provable."

$$\begin{aligned} x, y \in D \quad &\text{implies} \quad x \cdot y \in D, \\ x, (x \Rightarrow y) \in D \quad &\text{implies} \quad y \in D, \\ (x \Leftrightarrow y) \in D \quad &\text{implies} \quad x = y, \end{aligned}$$

where $(x \Rightarrow y) = -^*(x \cdot -y)$ and $(x \Leftrightarrow y) = (x \Rightarrow y) \cdot (y \Rightarrow x)$ are the operations interpreting strict implication and strict equivalence in K. These closure conditions on D are intended to correspond to Lewis' deduction rules of adjunction, strict detachment, and substitution of strict equivalents.

A matrix is a special kind of algebra. An algebra is a matrix without a set D of designated elements. Boolean algebras correspond to matrices for propositional logic. According to Bull and Segerberg [BS84] the generalization from matrices to algebras may have had the effect of encouraging the study of these structures independently of their connections to logic and modal systems. The set of designated elements D in fact facilitates a definition of validity with respect to which the theorems of a system can be evaluated. Without such a set the most obvious link to logic is severed. A second generalization to classes of algebras, rather than merely to individual algebras, was also crucial to the mathematical development of the subject matter. Tarski is the towering figure in such development.

Jónsson and Tarski [JT51, JT52] introduce the general idea of *Boolean algebras with operators*, i.e., extensions of Boolean algebras by addition of operators that correspond to the modal connectives. They prove a general representation theorem for Boolean algebras with operators that extends Stone's result for Boolean algebras (every Boolean algebra can be represented as a set algebra). This work of Jónsson and Tarski evolved from Tarski's purely mathematical study of the algebra of relations, and includes no reference to modal logic or even logic in general. Jónsson and Tarski's theorem is a more general algebraic analog of Kripke's later semantic completeness results, yet this was not realized for some time.

Relational Semantics

Key ideas surrounding relational interpretations of modality had occurred to several people. In the early 1940's the recognition of the semantical nature of the notion of logical truth led Rudolf Carnap to an informal explication of this notion in terms of Leibnizian possible worlds. At the same time, he recognized that the many syntactical advances in modal logic from 1918 on were still not accompanied by adequate semantic considerations. Carnap's work in the early forties [Car46, Car47] was focused on defining a formal semantic notion of L-truth to represent the informal semantic notions of logical truth.

Carnap introduces the apparatus of state-descriptions to define the formal semantic notion of L-truth. This formal notion is then to be used to provide a formal semantics for S5. A *state-description* for a language L is a class of sentences of L such that, for every atomic sentence p of L, either p or $\neg p$, but not both, is contained in the class. If S is a collection of state descriptions, and $s \in S$, then a propositional symbol p is satisfied at s if and only $p \in s$. Boolean operators are interpreted in the obvious way. As for the modal case, $\Box\alpha$ is satisfied at $s \in S$ if and only if for all $s' \in S$, s' satisfies α. Carnap's notion of validity or L-truth is a maximal notion, i.e., Carnap now defines a sentence to be valid or L-true if and only if it holds in all state-descriptions.

The standard relational semantics for modal logic is based on *possible worlds semantics*. Carnap's semantics is indeed a precursor of the possible worlds approach. Yet some crucial ingredients are still missing. First, the maximal notion of validity

must be replaced by a new universal notion. Second, definite descriptions must make space for possible worlds understood as indices or points of evaluation. Last, a relation of accessibility between worlds needs to be introduced. Though Kripke is by no means the only logician in the fifties and early sixties to work on these ideas, it is in Kripke's version of possible worlds semantics that all these innovations are present. Kanger [Kan57], Montague [Mon60], Hintikka [Hin61], and Prior [Pri62] were all thinking of a relation between worlds, and Hintikka [Hin61] like Kripke [Kri59] adopted a new notion of validity that required truth in all arbitrary sets of worlds. The intuitive idea behind the possible worlds model is that, besides the true state of affairs, there are a number of other possible states of affairs, or "worlds." Necessity then means truth in all possible worlds. Kripke's abstract characterization of the worlds is crucial in the provision of a link between the model theoretic semantics and the algebra of modal logic. Kripke saw very clearly this connection between the algebra and the semantics, and this made it possible for him to obtain model theoretic completeness and decidability results for various modal systems in a systematic way. Possible worlds semantics reaches its current form with Kripke in [Kri63], which explains why the mathematical structures that capture the possible worlds approach are called *Kripke structures*. It is due to Kripke's innovations that we now have a model theory for intensional logics.

Providing modal logics with a model theory was a revolutionary achievement. In the short span of time of less than fifty years, from Lewis' pioneering work starting in 1918 to Kripke's work in the early 1960's, modal logic flourished: different modal systems were developed and advances in algebra helped to foster the model theory for such systems. This culminated in the development of a formal semantics that extended to modal logic the successful first-order model theoretic techniques, thereby affording completeness and decidability results for many systems. And all these developments, along with the adoption of modal logic by theoretical computer science, helped to shift the view of modal logics as "intensional" formalisms that were only able to talk about "modes of truth" to a much broader panorama, which constitutes the current way of looking at modal logics.

1.2 On Modal Logics and Dynamic Logics

Nowadays modal logics [BdRV01, BvB06] can be thought of as a family of languages for talking about *structures* or *models*. But what kind of structures? As you might guess, there is no single answer to this question. For example, modal logic can be given an *algebraic semantics*, and under this interpretation modal logic is a tool for talking about Boolean algebras with operators; it can also be given a *topological semantics*, so modal logic can also be view as a tool for talking about topologies. Although there are alternative semantics, modal logic is now broadly used as a tool for talking about *graphs*. This view of modal logic uses relational structures, often called *Kripke structures*, and is the best known and best explored style of modal semantics. It is also, arguably, the most intuitive. *Relational semantics* has been used as a tool for reasoning about time, beliefs, computational systems, necessity and possibility, and much else besides. All these very different areas have in common that the fundamental concepts they need to model can be expressed in terms of graphs-like structures. And this is fortunate as many situations can be modeled using graphs: flow of time, relations between epistemic alternatives, transitions between computational states, networks of possible worlds, etc. This explains why modal logics have been used in many, diverse fields. Moreover, the range of modal logics

known today is extremely wide, so that it is usually possible to pick and choose the right modal logic for a particular application.

1.2.1 Basic Modal Logic

It is now time to formally meet the basic modal logic ($\mathcal{ML}$) and its relational semantics. We start by defining the syntax.

Definition 1.2.1 (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 $\varphi, \psi \in \mathsf{FORM}$. The necessity operator $\Box\varphi$ is a shorthand for $\neg\Diamond\neg\varphi$. Other operators are defined as usual: $\top$ is $\neg\bot$, $\varphi \vee \psi$ is defined as $\neg(\neg\varphi \wedge \neg\psi)$, $\varphi \to \psi$ is defined as $\neg\varphi \vee \psi$, and $\varphi \leftrightarrow \psi$ is an abbreviation for $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

Now that we have described the syntax of our language (that is, the set of well-formed formulas), we need to define the models in which we evaluate modal formulas. Models are just labeled directed graphs, which we call *Kripke models*.

Definition 1.2.2 (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. Informally we think of $V(p)$ as the set of points in $\mathcal{M}$ where p is true.

Since we already have both the syntax and the structures the language is going to talk about, we are now ready to define the semantics, that is, we are now able to tell whether a given formula is true or false in a model. Modal logics *describe* Kripke structures from an *internal* perspective. This means that modal formulas are evaluated at some particular point of the model. For this purpose we use pointed models: pairs of the form $(\mathcal{M}, w)$, where w is a state in $\mathcal{M}$; we usually drop parentheses and call $\mathcal{M}, w$ a pointed model.

Definition 1.2.3 (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

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

A formula φ is *satisfiable* if there exists a pointed model $\mathcal{M}, w$ such that $\mathcal{M}, w \models \varphi$. A formula φ is *globally satisfiable* in a model $\mathcal{M}$ if it is satisfied at all points in $\mathcal{M}$, and if this is the case we write $\mathcal{M} \models \varphi$. A formula φ is *valid* if it is globally satisfied in all models, and if this is the case we write $\models \varphi$.

The binary relation R is intended to capture the possibility relation: $(w, v) \in R$ if the state v is possible given the information in the state w. We think of R as a possibility relation, since it defines what states are considered possible in any given state. Note that $\Diamond\varphi$ is the possibility operator with respect to R, and its dual $\Box\varphi$ is the necessity operator with respect to R, which obtains the following truth condition.

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

Figure 1.1 shows an example of a Kripke model. As we can see, the model $\mathcal{M}$ is a graph with three elements, $\{w, v, u\}$. State w is labeled by p, state u is labeled by p and q, and state v has no label. Formally $\mathcal{M}{=}\langle W, R, V\rangle$, where $W{=}\{w, v, u\}$, $R{=}\{(w, v), (v, v), (v, u), (u, v), (u, w)\}$, and $V(p){=}\{w, u\}, V(q){=}\{u\}$.

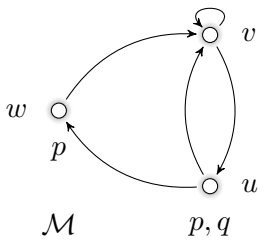


Figure 1.1: Example of a Kripke model.

Note that whether or not a formula is satisfied depends on where it is evaluated. For example, in Figure 1.1, the formula $\Diamond p$ is satisfied at points u and v (because at each state there is a successor point in which p is true) but not at w. Thus $\mathcal{M}, u \models \Diamond p$ because at u there is $w \in W$ such that $(u, w) \in R$ and $\mathcal{M}, w \models p$, and similarly, $\mathcal{M}, v \models \Diamond p$ because at v there is $u \in W$ such that $(v, u) \in R$ and $\mathcal{M}, u \models p$. But $\mathcal{M}, w \not\models \Diamond p$ because at w there is no accessible point in which p holds. Following the same reasoning, you can verify that $\mathcal{M}, v \models \Diamond(p \wedge q)$ and $\mathcal{M}, w \models \Diamond\Box\Diamond p$.

A few remarks are worth highlighting. First, note the *internal* character of the modal satisfaction definition. Modal formulas talk about Kripke models from *inside*. In first-order classical logic this is not the case, when we talk about a model, we do so from the *outside*. Second, note that modal logics *describe* characteristics of relational structures. Given a pointed model, the $\Diamond$ operator moves the evaluation of the formula in its scope to some successor of the evaluation point. In this way, it is possible to *describe* the model by traversing its structure.

Now that we have presented $\mathcal{ML}$, we are interested in its computational behavior. There are two main computational problems associated with modal logic. The first problem is checking if a given formula is true in a given state of a given Kripke structure. This problem is known as the *model checking problem*. The second problem is checking if a given formula is true in some state of some Kripke structure. This problem is known as the *satisfiability problem*. Both problems are decidable in $\mathcal{ML}$. The model checking problem can be solved in polynomial time, while the satisfiability problem is PSPACE-complete.

The basic modal logic has better computational properties than first-order logic. First-order logic ($\mathcal{FOL}$) is a well understood and powerful language [EFT96, End01, Smu95]. But from a computational point of view, it is sometimes too expressive. Its satisfiability problem is undecidable, while its model checking problem is PSPACE-complete. Table 1.1 summarises the computational properties of both logics.

	Model Checking	Satisfiability
$\mathcal{FOL}$	PSPACE-complete	Undecidable
$\mathcal{ML}$	Polinomial Time	PSPACE-complete

Table 1.1: Computational behavior of first-order logic and modal logic.

From Definition 1.2.1 it is clear that $\mathcal{ML}$ can be seen as an extension of propositional logic. But we can also show that $\mathcal{ML}$ is a fragment of fist-order logic. In particular, it is a fragment of first-order logic with two variables ($\mathcal{FOL}^2$), which

is decidable [Sco62]. Actually, its satisfiability problem is NEXPTIME-complete as shown by Grädel in [GKV97]. Intuitively, the states in a Kripke structure correspond to domain elements in a relational structure, and modalities are nothing but a limited form of quantifiers. The benefits of that limitation is that modal logic is "so robustly decidable" [Var96].

For a number of reasoning tasks of interest, such as model checking and satisfiability, the basic modal language is computationally better behaved than the corresponding first-order language. Of course, this better computational behavior comes about because the basic modal language is not nearly as expressive as first-order logic: $\mathcal{ML}$ 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 $\mathcal{FOL}^2$.) Thus the pressing questions are: what are are the trade-offs? And can this better computational behavior be lifted to more expressive modal logics? We shall try to discuss the answers to these questions in this thesis. We will introduce *dynamic* logics, as extensions of the basic modal language, which include operators that can alter the accessibility relation of a model during the evaluation of a formula, and we will investigate one important computational problem: the satisfiability problem. We will see that we obtain very expressive modal languages. In fact, if we imagine modal logic as a small boat navigating somewhere on the border between decidability and undecidability, we will see that if we add dynamic operators to modal logic we come to cross the border and we get to navigate on the undecidability side.

1.2.2 Dynamic Logics

We have seen that $\mathcal{ML}$ allow us to *describe* properties of a model by traversing its structure. In some sense, this is a *dynamic* behavior. Each time that we evaluate $\Diamond\varphi$, we scan for R-accessible points from the current one in search of one where φ is satisfied, and we move evaluation over there. As we move the evaluation of φ to a different point, we evaluate φ in a different pointed model, and it is precisely this behavior which allow us to describe properties of models. But this dynamic power seems limited as models never change after the application of certain operations. If we want to reason about dynamic phenomena it would be interesting to be able to *change* the structure.

Before moving on we should clarify our notion of dynamic logic, because some modal operators have been devised in the past to model dynamic phenomena, but not in the sense we just mentioned. One classical example is *propositional dynamic logic*, or PDL [Lad77, FL79, Har00]. PDL is a formal system for reasoning about programs. Originally, it was designed to formalize correctness specifications and to 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, then π^* is a program that executes π a finite number (possibly zero) of times.
- **Test:** if φ is a formula, then $\varphi?$ is a program that tests whether φ holds, and if so, continues, otherwise 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,

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

where R_π is either 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 (note that it goes beyond first-order logic, as it can express the reflexive-transitive closure of a relation while first-order logic can't), 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 behavior 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 behavior. PDL is dynamic in the sense of representing executions of programs, modeling the changes of a state after the application of an action. However, it never changes the model. In this thesis we are interested in operators that can change the model during the evaluation of a formula. There are many situations in which these kind of operators are used. Operators that change the accessibility relation are appropriate to model scenarios such as changes in the knowledge of an agent in epistemic logics. We will introduce *dynamic epistemic logics* in the next section to show the kind of dynamic behavior we are interested in.

1.3 Dynamic Epistemic Logic

We now move on to a different form of dynamics related to the topic of this thesis. Starting from the perspective of *epistemic logic* (the modal logic of *knowledge*), we can extend it with dynamic modal operators to model *change of knowledge*. The result is known as *dynamic epistemic logic* ($\mathcal{DEL}$). The starting point of $\mathcal{DEL}$ is therefore the logic of knowledge. Let us start with this.

1.3.1 Reasoning about Knowledge

Hintikka's book [Hin62] was one of the first works that, following von Wright's ideas on modal logic [vW51], formalized the concepts of knowledge and belief. Possible

world semantics [Kri63] resulted fruitful to interpret diverse notions such as temporal, dynamic, doxastic, deontic, and in particular, *epistemic reasoning* [FHMV04]. 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, and accessibility is represented by edges between states.

Let us define formally the syntax of the basic epistemic logic $\mathcal{EL}$.

Definition 1.3.1 (Syntax of Basic Epistemic Logic)**.** Let PROP be a countable, infinite set of propositional symbols and AGT a finite set of agent symbols. Then the set FORM of formulas of $\mathcal{EL}$ over PROP and AGT is defined as:

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

where $p \in \mathsf{PROP}$, $a \in \mathsf{AGT}$, and $\varphi, \psi \in \mathsf{FORM}$. Other operators are defined as usual.

$\mathcal{EL}$ extends the propositional language with an unary operator for each agent. $K_a\varphi$ is interpreted as "agent a knows that φ." Here, an agent may be a human being, a player in a game, a robot, a machine, or simply a "process." We also define the dual operator $\hat{K}_a\varphi$ as $\neg K_a\neg\varphi$. The fact that a does not know that $\neg\varphi$ is pronounce as "the agent considers it possible that φ." Hence, examples of well-formed formulas are $p \wedge \neg K_a p$ ("p is true but agent a does not know it"), and $\neg K_b K_c p \wedge \neg K_b \neg K_c p$ (saying that "agent b does not know *whether* agent c knows p"). Sometimes we refer to *modalities* instead of agents.

We now move on to the formal semantics, first defining the models in which we evaluate epistemic formulas.

Definition 1.3.2 (Epistemic Models)**.** An epistemic model is a Kripke model $\mathcal{M} = \langle W, \{R_a\}_{a\in\mathsf{AGT}}, V\rangle$, where each R_a is a binary relation on W that is reflexive, symmetric and transitive (an equivalence relation).

Epistemic models are Kripke models with multiple accessibility relations, one for each agent in the language. 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 1.3.3 (Semantics of Basic Epistemic 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

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

The clause for K_a is also phrased as "K_a is the necessity operator with respect to R_a." Note that the dual $\hat{K}_a$ obtains the following truth condition, for which it is also dubbed "a possibility operator with respect to R_a."

$$\mathcal{M}, w \models \hat{K}_a\varphi \text{ iff } \text{ for some } v \in W \text{ s.t. } (w, v) \in R_a,\ \mathcal{M}, v \models \varphi.$$

A crucial aspect of the semantics of $\mathcal{EL}$ is that it uses a special case of Kripke models. In such models, the notion of *indistinguishability* is of main importance. We explain this with an example.

Suppose that there is an agent Bob (represented by the agent symbol b) who lives in Paris. For some reason, he builds a theory about the weather conditions in both Paris and Rome: it is either sunny in Paris (represented by the propositional symbol p) or not ($\neg p$), and it is sunny in Rome (represented by r) or not ($\neg r$). 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 represented by model $\mathcal{M}$ in Figure 1.2. Since Bob lives in Paris, we can assume that he is aware of the weather in Paris, but not of that in Rome. In other words: he cannot distinguish state $\{p, r\}$ from $\{p, \neg r\}$, neither he can tell the difference between $\{\neg p, r\}$ and $\{\neg p, \neg r\}$.

Indistinguishability of agent b is represented by an edge labeled with b. An edge from a state w to v labeled with an agent symbol b is read as "in state w, agent b considers it possible that the state in fact is v," or "agent b cannot distinguish between states w and v." The latter 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 w and v are the same for agent b, and so are v and u, then b cannot distinguish between w and u. Note that the accessibility relation in model $\mathcal{M}$ of Figure 1.2 is indeed an equivalence relation.

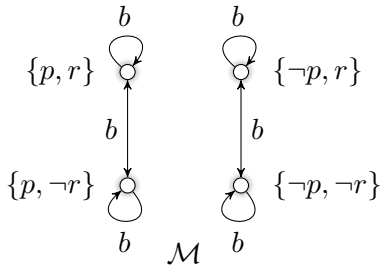


Figure 1.2: Epistemic model representing the weather scenario.

1.3.2 Changing Knowledge

We consider as information something that is relative to a subject (the agent) who has a certain perspective of 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 values of propositions describing the facts of the world that are independent of the agents, remain unchanged. *Dynamic epistemic logic* [vDvdHK07] is the field which studies this kind of information change, as an extension of the basic epistemic logic.

Several languages have been defined to represent information change. We will discuss the most basic one known as *public announcement logic* ($\mathcal{PAL}$). $\mathcal{PAL}$ was introduced in [Pla07] as an extension of $\mathcal{EL}$ with the operator $[!\psi]$ which communicates some common information to the agents.

Definition 1.3.4 (Syntax of Public Announcement Logic)**.** Let PROP be a countable, infinite set of propositional symbols and AGT a finite set of agent symbols. Then the

set FORM of formulas of $\mathcal{PAL}$ over PROP and AGT is defined as:

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid K_a\varphi \mid [!\psi]\varphi,$$

where $p \in \mathsf{PROP}$, $a \in \mathsf{AGT}$, and $\varphi, \psi \in \mathsf{FORM}$. Other operators are defined as usual.

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, i.e, the announcement is public, and then φ is evaluated. The dual of $[!\psi]$ is $\langle!\psi\rangle$. The formula $\langle!\psi\rangle\varphi$ therefore stands for "after *some* truthful public announcement of ψ, it holds that φ."

The effect of the public announcement of ψ is the restriction of the epistemic state to all states where ψ holds, including access between states. So, "announce ψ" can be seen as an epistemic state modifier, with a corresponding dynamic modal operator $[!\psi]$. We need to add a clause for the interpretation of such dynamic operator to the semantics:

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

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 changed 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. Note 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 abstract example from [vDvdHI12] to explain how public announcements work.

Consider model $\mathcal{M} = \langle W, \{R_a, R_b\}, V\rangle$ of Figure 1.3[3] modeling the uncertainty about agents a and b, where $W = \{w_1, w_2, w_3, w_4\}$, where R_a is the reflexive closure of $\{(w_3, w_4), (w_4, w_3), (w_1, w_2), (w_2, w_1)\}$, and where, similarly, agent b cannot distinguish w_2 from w_3 nor w_1 from w_4. We also have $V(p) = \{w_1, w_2\}$ and $V(q) = \{w_1, w_4\}$. Then,

$$\mathcal{M}, w_1 \models p \wedge q \wedge \neg K_a q \wedge \neg K_b p \wedge \hat{K}_a\hat{K}_b(\neg p \wedge \neg q)$$

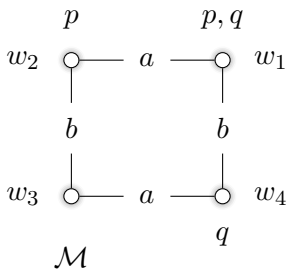


Figure 1.3: An epistemic model.

[3]We have slightly changed our graphical notation for the most commonly used in the epistemic logic field. As we know epistemic models involve equivalence relations, reflexive and transitive edges are ommitted, and symmetry is implicity represented by drawing undirected lines between states.

Now consider the announcement $p \vee q$: this changes model $\mathcal{M}$ to model $\mathcal{M}_1 = \mathcal{M}_{|(p \vee q)}$ as illustrated in Figure 1.4. The following is true in the pointed model $\mathcal{M}, w_1$ since (1) $p \vee q$ is true in w_1, and (2) the formula bounded by the announcement $\langle !p \vee q \rangle$ is true in $\mathcal{M}_1, w_1$:

$$\mathcal{M}, w_1 \models \langle !p \vee q \rangle (\neg K_a q \wedge \neg K_b p \wedge \neg \hat{K}_a \hat{K}_b (\neg p \wedge \neg q))$$

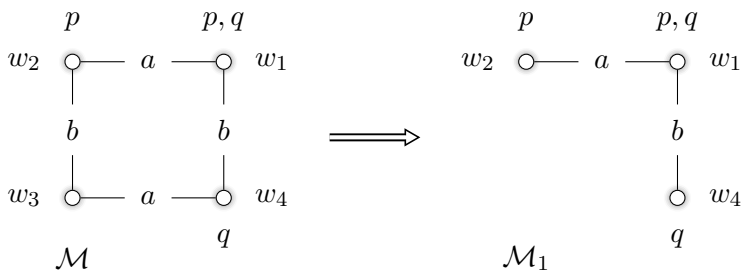


Figure 1.4: Changing model $\mathcal{M}$ to $\mathcal{M}_1$ after an announcement.

Note that the knowledge of the agents has changed (the formula $\hat{K}_a \hat{K}_b (\neg p \wedge \neg q)$ no longer holds in $\mathcal{M}_1, w_1$). Now suppose that in $\mathcal{M}_1, w_1$ agent a now publicly announces the true statement that he does not know q. Since in $\mathcal{M}_1, w_4$ agent a *does* know q (because there is a reflexive edge labeled by a), this state gets eliminated from the model, resulting in model $\mathcal{M}_2 = \mathcal{M}_{1\,|\neg K_a q}$ as illustrated in Figure 1.5. In other words, one effect of agent a announcing he does not know that q, is that the knowledge of agent b has changed: he comes to know that p! We have:

$$\mathcal{M}_1, w_1 \models \langle !\neg K_a q \rangle (\neg K_a q \wedge K_b p)$$

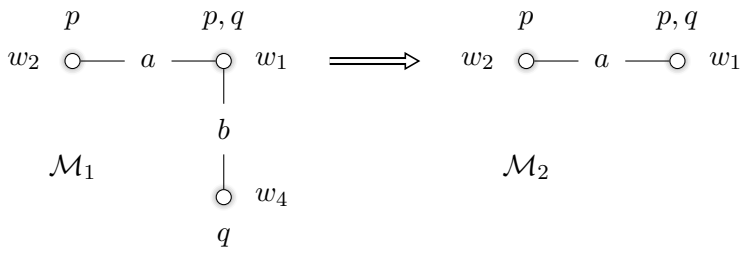


Figure 1.5: Changing model $\mathcal{M}_1$ to $\mathcal{M}_2$ after an announcement.

Finally, if in $\mathcal{M}_2, w_1$ agent b announces the true proposition $K_b q$, we end up in model $\mathcal{M}_3 = \mathcal{M}_{2\,|K_b q}$ as illustrated in Figure 1.6. Now both agents know the same information. So we have:

$$\mathcal{M}_2, w_1 \models \langle !K_b q \rangle (K_a(p \wedge q) \wedge K_b(p \wedge q))$$

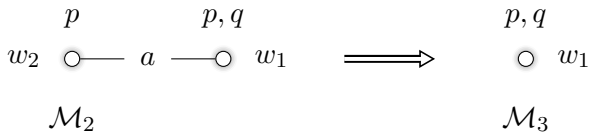


Figure 1.6: Changing model $\mathcal{M}_2$ to $\mathcal{M}_3$ after an announcement.

All in all, the three announcements can be made at once in $\mathcal{M}, w_1$, obtaining the sequence of model changes shown in Figure 1.7. That is,

$$\mathcal{M}, w_1 \models \langle !p \vee q \rangle \langle !\neg K_a q \rangle \langle !K_b q \rangle (K_a(p \wedge q) \wedge K_b(p \wedge q))$$

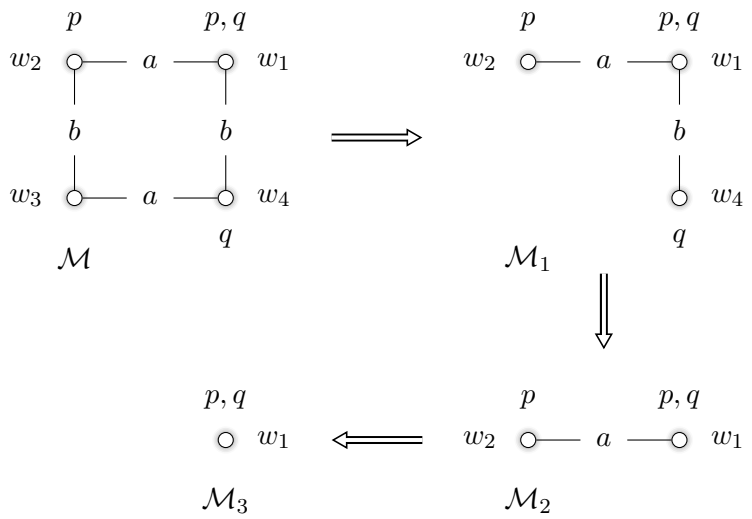


Figure 1.7: Changing model $\mathcal{M}$ after three consecutive announcements.

With this example it should be clear that information change is represented by *changing* models. In terms of Kripke models that means that the accessibility relations of the agents have to change (and consequently the set of states of the model might change as well). Intuitively, the idea is that if an agent considers fewer worlds possible, then he has less uncertainty, and therefore more knowledge.

The reasoning tasks we study in this chapter, the model checking problem and the satisfiability problem, are both decidable for $\mathcal{PAL}$. The model checking problem can be solved in polynomial time, while the satisfiability problem is PSPACE-complete. It is known that $\mathcal{PAL}$ and $\mathcal{EL}$ have the same expressive power, and in [Lut06] it is shown that the computational complexity of $\mathcal{PAL}$ coincides with that of $\mathcal{EL}$. This shows that adding a dynamic modality with public announcements to $\mathcal{EL}$ does not have negative effects on computational complexity. Table 1.2 summarises the computational properties of both logics.

	Model Checking	Satisfiability
$\mathcal{PAL}$	Polinomial Time	PSPACE-complete
$\mathcal{EL}$	Polinomial Time	PSPACE-complete

Table 1.2: Computational behavior of public announcement logic and epistemic logic.

The logic of public announcements is very simple and is a good example of the kind of dynamic behavior we are interested to investigate in this thesis. Although many situations can be modeled using $\mathcal{PAL}$, we are not particularly interested in the way in which we can represent change of knowledge. Instead, what is interesting to us is the the ability of changing the model. We are interested in changing the structure of the model by means of operators that can delete, add, and invert an edge in the accessibility relation during evaluation. In the next chapter we will introduce these dynamic operators as extensions of the basic modal logic, and we will see that we obtain more expressive logics, which in contrast to $\mathcal{PAL}$, do have negative effects on computational complexity. In fact, we will see that they turn out to be undecidable.

1.4 About This Thesis

The themes discussed in this introduction should give the reader an intuitive idea of the topic of this thesis. We started with a brief overview of the history of modal logic following the syntactic, algebraic, and model theoretic traditions, from Lewis' pioneering work starting in 1918 to Kripke's work in the early 1960's. Next we gave a more contemporary introduction of the subject matter by introducing the basic modal logic along with a discussion of its computational behavior. We then moved to the subject of dynamic logics. We briefly discussed about PDL to show that some logics are called "dynamic" but not in the sense of changing the structure of the model. Dynamic epistemic logics are better suited for this. We introduced public announcement logic emphasizing that it can model dynamic phenomena through changes in the model, and we discussed briefly about its computational behavior.

If you survived up to here, you are ready to go on. This introductory chapter was written aiming at presenting modal and dynamic logics mainly from a computational perspective, to lay a logical background before introducing the logics we are going to work with on this thesis. We will introduce a family of logics defined to represent dynamic behavior. In particular, we will investigate logics that can modify the accessibility relation of a model during the evaluation of a formula. We call this family *relation-changing logics*. We will introduce six relation-changing operators: *sabotage*, which deletes edges of the model; *bridge*, which adds new edges, and *swap*, which turns around edges. All of them are introduced in two versions: *local*, i.e, modifying adjacent edges from the evaluation point, and *global*, changing arbitrary edges of the model. This thesis is organized as follows:

In Chapter 2 we introduce the family of relation-changing logics. We first introduce them informally, using dynamic epistemic logic as a motivation, and we then present their formal syntax and semantics. By the end of the chapter we provide some examples to show their expressive power and we give a short overview of some known results on expressivity and computational complexity.

Chapters 3 and 4 form the core of this thesis. In Chapter 3 we study the finite model property and we show that relation-changing logics can force infinite models. To check this we use a spy point technique, a classical tool to prove expressiveness results in logic. The idea is to characterize a point (the spy point) which has a global view of the state of the model, and use it to describe the properties we want to impose in the model. In this way we show that our logics lack the finite model property, which is a key property to have decidability. This leads us to the central question of this thesis: have we added enough expressive power to cross the border of decidability? This indeed will be the case, and in Chapter 4 we provide undecidability results for the six logics we introduce. To obtain these results we reduce the undecidable satisfiability problem of a logic called *memory logic* to the satisfiability problem of each or our logics, using again the spy point technique to define the encodings.

Finally, we conclude in Chapter 5 giving a brief summary of this thesis and reporting on my own experience of working on it.

CHAPTER 2

Relation-Changing Logics

In this chapter we will introduce the logics we are going to work with on this thesis. We will focus in modal operators that can change the accessibility relation of a model. Other model changing operators, such as public announcement in $\mathcal{DEL}$, have been studied extensively, but arbitrarily *changing the access* seems appealing to investigate. 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 behavior.

2.1 Modal Logics over Changing Models

We have seen that epistemic logic is a modal logic for reasoning about knowledge and belief, and that even more interesting, it can be extended with dynamic operators to model information change. Dynamic epistemic logic is an umbrella term for a number of extensions of epistemic logic. For instance, public announcement logic adds the operator $[!\psi]$ which announces some truthful information that becomes common knowledge for the agents. This dynamic behavior is represented in the language through changes in the model (deleting edges and states).

In contrast, the basic modal logic defined in Section 1.2.1 does not allow us to reason about neither knowledge nor belief. It let us *describe* relational models in an abstract way without giving any particular interpretation to the symbols of the language. But why can a logic only *describe* a model? What about *changing* the model? If we want to reason about *dynamic* aspects of a given situation, e.g., how the relations between a set of elements evolve through time or through the application of certain operations, the use of logics with classical semantics becomes less clear. It would be interesting to model dynamic behavior in a similar manner to that of dynamic epistemic logics, i.e., by changing the model.

We will introduce dynamic logics which include operators that can alter the accessibility relation of a model during evaluation. As epistemic logic is the starting point of dynamic epistemic logics, the basic modal logic will be the starting point of our *relation-changing logics*.

Several relation-changing operators are interesting to us. The first one, *sabotage*, has the ability to delete an edge during the evaluation of a formula. The sabotage operator, $\langle \mathsf{sb} \rangle$, is a $\Diamond$-like operator (to be true at a state w it requires the existence of

an accessible state v where evaluation will continue) but it changes the accessibility relation during evaluation (the pair (w, v) is deleted). A picture will help understand the dynamics of $\langle \mathsf{sb} \rangle$. The formula $\langle \mathsf{sb} \rangle \Box \bot$ is true in a model with two related states:

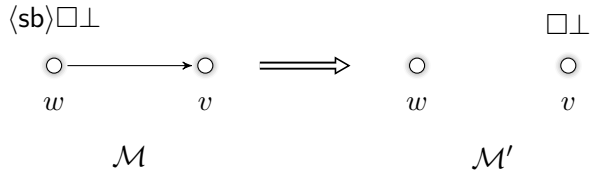


Figure 2.1: Changing model $\mathcal{M}$ to $\mathcal{M}'$ after evaluating the $\langle \mathsf{sb} \rangle$ operator.

As we can see in the picture, evaluation starts at state w with the arrow pointing from w to v, but after evaluating the $\langle \mathsf{sb} \rangle$ operator, it continues at state v with the arrow deleted. Note that states do not change, only the accessibility relation is changed. It is clear that the $\langle \mathsf{sb} \rangle$ operator *changes* the model in which a formula is evaluated.

The $\langle \mathsf{sb} \rangle$ operator is *local*, in the sense that it modifies adjacent edges from the evaluation point. We can also define a *global* operator, which allow us to change edges anywhere in the model, not just adjacent edges. This *global* sabotage operator, $\langle \mathsf{gsb} \rangle$, was introduced by Johan van Benthem in [vB05]. A picture will help understand the dynamics of $\langle \mathsf{gsb} \rangle$. The formula $\langle \mathsf{gsb} \rangle \Diamond \Box \bot$ is true in a model with three related states:

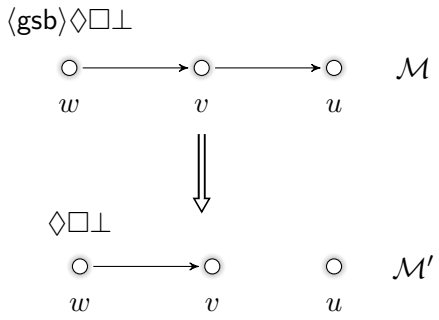


Figure 2.2: Changing model $\mathcal{M}$ to $\mathcal{M}'$ after evaluating the $\langle \mathsf{gsb} \rangle$ operator.

As we can see in the picture, evaluation starts at state w with an arrow pointing from w to v and another arrow pointing from v to u, but after evaluating the $\langle \mathsf{gsb} \rangle$ operator, it stays in the same state w with the arrow pointing from v to u deleted. Why? Well, the formula $\Diamond \Box \bot$ says that there is an accessible state which is a dead-end. At state w the only way that this formula holds is by deleting the arrow from v to u. Note that state u is not deleted, only the edge is deleted. And also note that after evaluating the $\langle \mathsf{gsb} \rangle$ operator evaluation stays in the same state as it was before: from the evaluation point we can change arbitrary edges of the model that need not be adjacent edges of the current point, expressing in this way *global* changes. You can think of a global modality as a universal modality. Sometimes global operators can be confusing, but things will become clearer when we present the semantics.

We will investigate four other dynamic operators in this thesis: $\langle \mathsf{br} \rangle$, for *bridge* (in its local version), which models the opposite situation of $\langle \mathsf{sb} \rangle$: it adds an arrow to an inaccessible state of the model and moves evaluation over there; $\langle \mathsf{gbr} \rangle$, which is the global counterpart of $\langle \mathsf{br} \rangle$; $\langle \mathsf{sw} \rangle$, for *swap* (in its local version), which has the

ability to invert the direction of a traversed arrow; and $\langle \mathsf{gsw} \rangle$, which is the global counterpart of $\langle \mathsf{sw} \rangle$. We add the *sabotage*, *bridge*, and *swap* operators to the basic modal logic, both in its local and global versions, and we obtain a family of six new logics that we call *relation-changing logics*.

We have chosen these relation-changing operators with the intention of covering a sufficiently varied sample of alternatives. Clearly, other operators could have been included in this exploration, and actually some alternative choices have been already investigated in the literature, e.g, the adjacent sabotage operator discussed in [Roh06].

2.2 Syntax and Semantics

It is now time to formally meet the relation-changing logics and its relational semantics. As usual, we start by defining the syntax.

Definition 2.2.1 (Syntax of Relation-Changing Logics)**.** Let PROP be a countable, infinite set of propositional symbols. Then the set FORM of formulas over PROP is defined as:

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

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

Formulas of the basic modal language $\mathcal{ML}$ are those that contain only the $\Diamond$ operator besides the Boolean operators. For $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle\}$ we call $\mathcal{ML}(\blacklozenge)$ the extension of $\mathcal{ML}$ allowing also the $\blacklozenge$ operator.

Formulas of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, $\mathcal{ML}(\langle \mathsf{br} \rangle)$, $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$, $\mathcal{ML}(\langle \mathsf{sw} \rangle)$, and $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ are evaluated in standard relational models, and the meaning of the operators of the basic modal logic remains 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 2.2.2 (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 $V : \mathsf{PROP} \mapsto \mathcal{P}(W)$ 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 $R_S^- = R \backslash S$, $S \subseteq R$.
(bridging)	$\mathcal{M}_B^+ = \langle W, R_B^+, V \rangle$, with $R_B^+ = R \cup B$, $B \subseteq (W \times W) \backslash R$.
(swapping)	$\mathcal{M}_S^* = \langle W, R_S^*, V \rangle$, with $R_S^* = (R \backslash S^{-1}) \cup S$, $S \subseteq R^{-1}$.

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 in the accessibility relation by dynamic operations. This notation will be a practical form to present the semantics of the new operators.

Definition 2.2.3 (Semantics of Relation-Changing Logics)**.** 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

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

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

Relation-changing operators can delete, add or swap around edges. The formula is then evaluated in the correspondent model variant. Note that we extend the basic modal logic with modalities that let us perform these operations in two versions: local and global. The local version sabotages, bridges or swaps edges moving evaluation to a successor point, while the global version modifies edges in any part of the model from the current evaluation point.

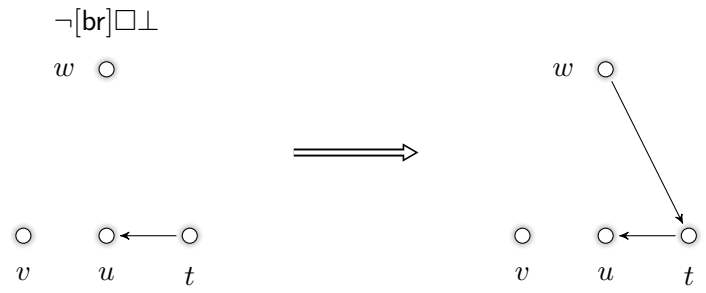
2.3 Some Examples

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 2.3.1. Besides deleting edges, the $\langle \mathsf{sb} \rangle$ operator has no effect on acyclic models (its behavior 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.

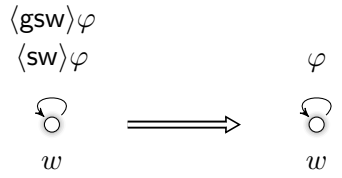
Example 2.3.2. This example is extracted from [Roh06], and shows the expressivity of $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$. Consider the formula $[\mathsf{gsb}]\Diamond\top \wedge \langle \mathsf{gsb} \rangle\langle \mathsf{gsb} \rangle\Box\bot$. The first conjunct expresses that, if an arbitrary 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 2.3.3. The $\langle \mathsf{br} \rangle$ operator allows us to reach isolated parts of the model:

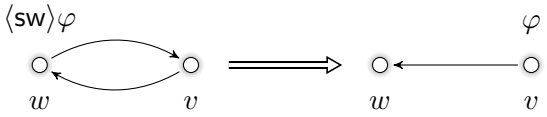


Starting from w, which is a point with no successors, by evaluating $\neg[\mathsf{br}]\Box\bot$ it is possible to reach an isolated part of the model and continue the evaluation of the rest of the formula. $[\mathsf{br}]\Box\bot$ establishes that all the points that have no links from w, have no successors. 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, checking that u is reachable from t).

Example 2.3.4. The $\langle \mathsf{sw} \rangle$ and $\langle \mathsf{gsw} \rangle$ operators leave reflexive edges unchanged:



Example 2.3.5. The $\langle \mathsf{sw} \rangle$ 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 $\langle \mathsf{sw} \rangle$-formula leads to a model variant where the size of the accessibility relation R decreases. The same happens with the $\langle \mathsf{gsw} \rangle$ operator, when it swaps a symmetric edge.

These examples show that logics with the ability to delete, add or swap edges of a model are quite expressive. The logics we presented in this chapter are first introduced in [AFH12] and are further studied in Fervari's PhD thesis [Fer14a]. In [AFH12, Fer12, Fer14a] it has been shown that adding any of the sabotage, bridge, or swap operators to the basic modal logic increases its expressive power: a suitable notion of bisimulation is defined, and the expressive power of the resulting logics is investigated, showing that they the lack the tree model property and that they are all incomparable. Moreover, in [AFH14] it is shown that the local version of swap logic is equivalent to a fragment of first-order logic.

Regarding its computational behavior, in [LR03a, LR03b] it is shown that the sabotage game is PSPACE-hard and that the model checking problem of the associated modal logic is PSPACE-complete. More recently, in [AFH12, Fer12, Fer14a] it has been proved that the model checking problem of the six dynamic logics we presented (sabotage, bridge, and swap, both in their local and global versions) is decidable and PSPACE-complete. Moreover, in [AFH13] tableaux methods for checking the satisfiability of relation-changing logics are defined. From [LR03a] we already know that the satisfiability problem of a variation of the global version of sabotage logic with multiple relations is undecidable, and similar results have been proved in [AFH14] for the local version of swap logic, but there is no certainty about the other logics. Table 2.1 summarises the known computational properties of relation-changing logics. In this thesis we will complete this panorama.

	Model Checking	Satisfiability
$\mathcal{ML}(\langle \mathsf{sb} \rangle)$	PSPACE-complete	?
$\mathcal{ML}(\langle \mathsf{gsb} \rangle)$	PSPACE-complete	Undecidable
$\mathcal{ML}(\langle \mathsf{br} \rangle)$	PSPACE-complete	?
$\mathcal{ML}(\langle \mathsf{gbr} \rangle)$	PSPACE-complete	?
$\mathcal{ML}(\langle \mathsf{sw} \rangle)$	PSPACE-complete	Undecidable
$\mathcal{ML}(\langle \mathsf{gsw} \rangle)$	PSPACE-complete	?

Table 2.1: Computational behavior of relation-changing logics.

With the known results on expresive power, model checking, and decidability it is natural to think that the all of the relation-changing logics are undecidable. These are the main results of this thesis: we will prove that the satisfiability problem of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, $\mathcal{ML}(\langle \mathsf{br} \rangle)$, $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$, $\mathcal{ML}(\langle \mathsf{sw} \rangle)$, and $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ is undecidable. Some of these results have already been published in [Fer14a] and [Fer14b].

Before moving on to the undecidability proofs itself, we will first show more evidence on the high expressive power of the logics: we will prove that adding relation-changing operators to the basic modal language allow us to enforce infinite models. This means that the logics are expressive enough to cross the border of decidability. This is the topic of the next chapter.

CHAPTER 3

Forcing Infinite Models

3.1 The Finite Model Property

In this chapter we will investigate the *finite model property*, a classical property of modal logic which establishes that every satisfiable formula is satisfied in a finite model. The basic modal language has the finite model property, or put it in another way, it does not have the expressive strength required to force the existence of infinite models. Let us introduce the formal definition.

Definition 3.1.1 (**Finite Model Property**)**.** Let $\mathcal{L}$ be a modal logic and $\mathcal{M}$ a set of finitely based models. We say that $\mathcal{L}$ has the *finite model property with respect to* $\mathcal{M}$ if the following holds: if φ is a formula of $\mathcal{L}$, and φ is satisfiable in some model in $\mathcal{M}$, then φ is satisfiable in a *finite* model in $\mathcal{M}$.

In fact, the basic modal language has a rather strong form of the finite model property: a formula is satisfied in a finite model whose size is bounded by a function on the size of the formula.

Definition 3.1.2 (**Strong Finite Model Property**)**.** Let $\mathcal{L}$ be a modal logic, $\mathcal{M}$ a set of finitely based models, and f a function mapping natural numbers to natural numbers. We say that $\mathcal{L}$ has the *strong finite model property with respect to* $\mathcal{M}$ if the following holds: if φ is a formula of $\mathcal{L}$, and φ is satisfiable in some model in $\mathcal{M}$, then there is a *computable* function f such that φ is satisfiable in a *finite* model in $\mathcal{M}$ containing at most $f(|\varphi|)$ states.

Then, we can introduce the following theorem for the basic modal language.

Theorem 3.1.3. *$\mathcal{ML}$ has the strong finite model property.*

Proof (Sketch). A complete proof via selection or filtration can be found in [BdRV01]. We explain in a few words the argument of the proof via filtrations. The filtration method for modal logics is based on the following 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 $2^{|\varphi|}$ states, thus we have a computable (though, unfortunately, exponential) upper bound for the size of the finite model. □

The finite model property is a characteristic feature of many modal logics. The strong finite model property provides a simple and semantic way of proving decidability. This is formalized in the following theorem.

Theorem 3.1.4. *If $\mathcal{L}$ is a modal logic that has the strong finite model property with respect to a recursive set of models $\mathcal{M}$, then $\mathcal{L}$ is decidable.*

Proof (Sketch). Suppose that modal logic $\mathcal{L}$ has the strong finite model property. Then for any formula φ there is a computable function f such that $f(|\varphi|)$ is an upper bound on the size of the models needed to satisfy φ. Now write a mechanical procedure that takes φ as input, generates all the finite models belonging to $\mathcal{M}$ up of size $f(|\varphi|)$, and tests for the satisfiability of φ on these models (note that as $\mathcal{M}$ is a recursive set we can indeed generate all such finite models). Because φ is satisfiable if and only if it is satisfied in a model of size $f(|\varphi|)$, and because the mechanical procedure systematically generates all these models, our mechanical procedure decides the satisfiability of $\mathcal{L}$. □

The finite model property can give us decidability of the logic: the strong finite model property gives a decision procedure itself, provided there is an effective way of recognizing finite models of the logic. But note that every modal logic that *does have* the finite model property is not necessarily decidable. On the other hand, if a modal logic *does not have* the finite model property, we cannot prove decidability via finite models. For logics lacking the finite model property it may also be possible to prove decidability results by computing with more abstract kinds of finite structure, such as quasi-models and mosaics (for an explanation of these structures see Chapter 6 of [BdRV01]). However, if we are working with high expressive modal languages that do not have the finite model property, undecidability can arise very easily.

The relation-changing logics we presented in Chapter 2 are very expressive because they not only let us describe models but also allow us to *change* them. Adding relation-changing operators to the basic modal language increases its expresive power, as shown in [AFH12, Fer14a]. The challenge is to discover how much expressivity we added. In other words, we want to know if the finite model property holds for these logics. As you might guess, relation-changing logics lack the finite model property. We will prove that the sabotage, bridge, and swap logics, both with local and global effects, can enforce formulas that hold in an infinite model. These are the first results of the thesis, and according to what we have discussed in the previous paragraph above, we can see that these results are the first signs of undecidability.

We introduce the following theorem.

Theorem 3.1.5. *$\mathcal{ML}(\blacklozenge)$ does not have the finite model property, for $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle\}$.*

To prove this theorem we need to show that each logic can enforce an infinite model. Informally, we will try to write an $\mathcal{ML}(\blacklozenge)$-formula which tell us that there is a non-empty set B of elements that forms a strict partial order, i.e., an irreflexive and transitive set, such that every element of the set has a successor. Note that this is very easy to specify with a first-order formula:

$$\forall x((x,x) \notin R) \ \wedge \ \forall x,y,z((x,y) \in R \wedge (y,z) \in R \rightarrow (x,z) \in R) \ \wedge \ \forall x \exists y((x,y) \in R)$$

Such a simple first-order formula only has infinite models. But the infinite models we want to enforce cannot be specified in such an easy way, they are a bit more difficult. Remember that, in modal logic, every formula is evaluated at a particular point of

the model, so we need to talk about a *set of points* that forms a strict partial order from a particular point in the model. But, how can we write an $\mathcal{ML}(\blacklozenge)$-formula that talks about a set of points with the desired properties that is going to be evaluated at a *unique point*? Well, we will use a *spy-point technique* [BS95]. A spy point is a state of the model which can access any other state in the model, and the idea is to use it to describe the partial order set we want to enforce in the model. We will use different kinds of spy points according to the expressivity of each language. This technique will allow us to write formulas to enforce serial, irreflexive, and transitive models, which implies that the models are infinite.

With these ideas in mind, we are now ready to present the formulas for enforcing infinite models. We will do so by presenting a formula for each of the sabotage, bridge, and swap logics, both in their local and global versions. We start with sabotage logic.

3.2 Sabotage Logic

3.2.1 Local Sabotage

We exhibit a formula φ, which is a conjunction of several properties:

$$
\begin{aligned}
\varphi = \quad & s \wedge \Box\neg s \wedge \Diamond\top \wedge \Box\Diamond s && (1)\\
& \wedge \Box\Box(s \to \Box\neg s) && (2)\\
& \wedge [\mathsf{sb}][\mathsf{sb}](s \to \Box\Diamond s) && (3)\\
& \wedge \Box[\mathsf{sb}](s \to \Diamond\neg\Diamond s) && (4)\\
& \wedge \Box\Diamond\neg s && (5)\\
& \wedge \Box\Box(\neg s \to \Diamond(s \wedge \neg\Diamond s)) && (6)\\
& \wedge \Box[\mathsf{sb}](\neg s \to [\mathsf{sb}](s \to \Box\Box(\neg s \to \Diamond s))) && (7)\\
& \wedge \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\Diamond(\neg s \wedge \neg\Diamond s))) && (8)\\
& \wedge \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\neg\Diamond s)) && (9)\\
& \wedge \Box[\mathsf{sb}](s \to \Box(\neg\Diamond s \to \Box\Diamond s)) && (10)\\
& \wedge \neg\Diamond\langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \Diamond s \wedge \Diamond\neg\Diamond s))) && (11)\\
& \wedge \Box\langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\neg\Diamond s))))) && (12)
\end{aligned}
$$

The intended model is illustrated in Figure 3.1.

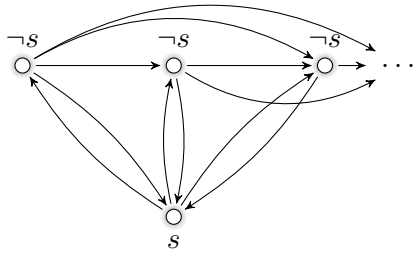
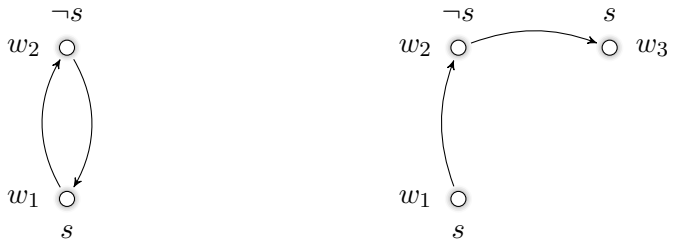


Figure 3.1: Infinite model for $\mathcal{ML}(\langle\mathsf{sb}\rangle)$.

Intuitively, the idea is that we evaluate φ at a unique point in which the propositional symbol s is true: this is the spy point. We force the existence of arrows going from the spy point to every successor, and arrows going from every successor back to the spy point. Formulas (1) to (9) are mainly devoted to this task, in addition of enforcing an infinite chain of states. We choose to define such a spy point because $\mathcal{ML}(\langle\mathsf{sb}\rangle)$ can delete arrows: by sabotaging some of the arrows that are connected to the spy point we can identify the visited states by simply checking if an arrow has

been deleted. In this way we can enforce an infinite chain of states that is irreflexive (formula (10)) and transitive (formulas (11) and (12)).

Let us now see with some level of detail how each part of φ works to get the intended model. The first half of (1) establishes that s is true at the evaluation point and that $\neg s$ is true at every successor accessible from there (in particular, it says that the evaluation point cannot be reflexive). The second half of (1) ensures that there is at least one successor (making the set of states the spy point is going to talk about not empty), and that each successor reaches an s-point. Note that the last part of (1), namely $\Box\Diamond s$, says that each successor reaches a point where s is true, but this point is not necessarily the evaluation point: it could be any other point where s might be true, as the following figure shows:



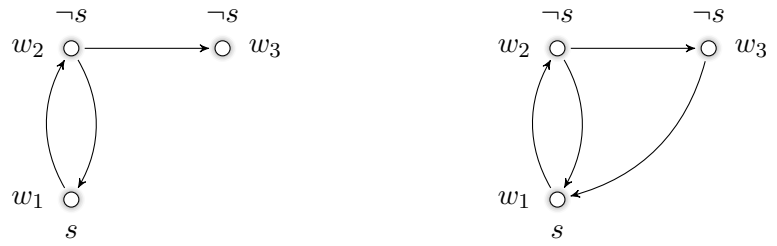
(a) Intended model (b) An unwanted model

Figure 3.2: Possible models for formula (1)

Formulas (2), (3) and (4) add constraints to enforce arrows back to the evaluation point, thus making it *unique*. These formulas do their job by distinguishing the evaluation point from any other s-point reachable in two steps. In particular, formulas (2) and (3) establish that $\Box\neg s$ and $\Box\Diamond s$ (two conditions that hold in the evaluation point) also hold in any other s-point reachable within two steps, but in addition to that, formula (3) deletes the traversed arrows. These changes in the model let us define the distinguishing property: the evaluation point always have a cycle of size two while any other s-point reachable within two steps cannot have such a cycle.[1] Then formula (4) verifies the property we have just enforced: if we have a cycle of size two, we can traverse an arrow, delete the other, and then check for the deleted arrow in order to know if we have returned to the point where we started. As we know that the evaluation point is the only s-point reachable in two steps that can have a cycle, any arrow that goes to another s-point different from the evaluation point is not consider in the model, because does not satisfy (4). In this way, though somewhat complicated, we get arrows back to the evaluation point as we wanted in the first place.

Formula (5) allow us to creates an infinite chain of elements by ensuring that each successor has an arrow to a point where $\neg s$ is true. The intended model is illustrated in the left hand side of Figure 3.3. Then formulas (6), (7) and (8) play the same role as (2), (3) and (4) but now enforcing an arrow back to the evaluation point for states accessible in two steps, as shown in the right hand side of Figure 3.3. Formula (9) makes the evaluation point a "spy point": any point accessible in two steps from the evaluation point can also be accessible in one. In Figure 3.3b this means that it forces the existence of an arrow going from w_1 to w_3, thus making w_3 a successor of w_1. But then if w_3 is a successor, in particular satisfies formula (5),

[1] In Figure 3.2a we can see that we have a cycle between w_1 and w_2, while in Figure 3.2b we cannot have such a cycle between w_2 and w_3: this is the property that distinguishes w_1 from w_3. In particular, we are saying that there cannot be an arrow from w_3 to w_2.



(a) Intended model for (5) (b) Intended model for (6), (7) and (8)

Figure 3.3: Intended model for formulas (5), (6), (7) and (8)

meaning that there must be an arrow from w_3 to another point, call it w_4, such that $\neg s$ is true; and w_4 in particular satisfies formulas (6), (7) and (8), meaning that w_4 must have an arrow to w_1; and w_4 also satisfies formula (9), meaning that w_4 is now a successor of w_1; but as w_4 is now a successor point, then it must have an arrow to a point w_5 satisfying the previous formulas. This process goes on forever, creating an infinite chain of states where the spy point can access any other point in the model.

Note that it is possible for formula (5) to enforce reflexive arrows. So in order to obtain an infinite model it remains to ensure that the set of states the spy point is talking about is irreflexive and transitive. Formula (10) enforces irreflexivity: after going to any successor and deleting the arrow that goes to the spy point (identifying in this way the point we are visiting), all the successors of the visited state can only see a state that satisfies s (the spy point). If the state were reflexive, after doing one loop it wouldn't be possible to reach the spy point, because we deleted the arrow to access it. Hence the visited state has to be irreflexive. Formula (12) enforces transitivity: if from one point we have an arrow to a second point, and from the second we have an arrow to a third, then we have an arrow from the first point to the third. We achieve this by identifying the first and third points, i.e, by deleting the corresponding arrows that go to the spy point, and then we enforce an arrow from the first to the third point, given that we have a point in the middle that acts as a bridge. By the way in which we do this, it is possible to enforce an arrow that goes from the third point to the first, something we clearly don't want. To avoid this situation there is formula (11) to provide the needed help: it ensures that there are no three elements of the infinite chain of states that form a cycle (again, we do it by deleting some arrows). Thus formulas (11) and (12) together enforce a transitive set of points.

Proposition 3.2.1. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model and $s \in W$ a point of the model. If $\mathcal{M}, s \models \varphi$ then $\mathcal{M}$ is infinite.*

Proof. Suppose $\mathcal{M}, s \models \varphi$. Let $B = \{w \in W \mid (s, w) \in R\}$. Because (1) is satisfied, $s \notin B$, $B \neq \emptyset$, and for all $w \in B$ there is $w' \in W$ such that $(w, w') \in R$ and $w' \in V(s)$. As (2), (3), and (4) are satisfied, $w' = s$, meaning that for all $w \in B$, $(w, s) \in R$. As (5) is satisfied at s, every point in B has an R-successor distinct from s, and because (6), (7), and (8) are satisfied, such R-successor of B has an edge to s. Because (9) is satisfied, if $w \neq s$ and w is an R-successor of an element of B then w is also an element of B. As (10) is satisfied at s, every point in B is irreflexive; and because (11) and (12) are satisfied, R transitively orders B. Hence B is an unbounded strict partial order, thus B is infinite and so is $\mathcal{M}$. □

Proposition 3.2.2. *There is an infinite model $\mathcal{M}$ and a point s such that $\mathcal{M}, s \models \varphi$.*

Proof. Let $(\mathbb{N}, <)$ be the natural numbers in their usual order, and suppose $s \notin \mathbb{N}$. Let $\mathcal{M} = \langle W, R, V \rangle$ be a model where W is $\mathbb{N} \cup \{s\}$, R is $< \cup \{(s,n),(n,s) \mid n \in \mathbb{N}\}$, and V is any valuation such that $V(s) = \{s\}$ and $V(n) \cap V(s) = \emptyset$, for $n \in \mathbb{N}$. It is clear that $\mathcal{M}, s \models \varphi$. Thus φ has at least one infinite model. □

3.2.2 Global Sabotage

We exhibit a formula φ, which is a conjunction of several properties:

$$
\begin{aligned}
\varphi = \quad & s \wedge \Box\neg s \wedge \Diamond\top && (1)\\
& \wedge \Box(\Diamond s \wedge \langle \mathsf{gsb} \rangle \neg\Diamond s) && (2)\\
& \wedge \Box\Diamond\neg s && (3)\\
& \wedge \Box\Box(\neg s \to \Diamond s \wedge \langle \mathsf{gsb} \rangle \neg\Diamond s) && (4)\\
& \wedge [\mathsf{gsb}](\Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \neg\Diamond s)) \;\to\; \Diamond\neg\Diamond s) && (5)\\
& \wedge [\mathsf{gsb}]\Box(\neg\Diamond s \to \Box\Diamond s) && (6)\\
& \wedge [\mathsf{gsb}]\neg\Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \Diamond s \wedge \Diamond\neg\Diamond s)) && (7)\\
& \wedge [\mathsf{gsb}][\mathsf{gsb}](\Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \neg\Diamond s)) \;\to\; \Diamond(\neg\Diamond s \wedge \Diamond\neg\Diamond s)) && (8)
\end{aligned}
$$

We enforce an infinite model in a similar way to that of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, except that now the s-successors in two steps may be different from the spy point, as illustrated in Figure 3.4.

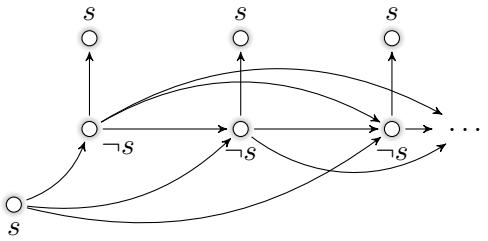


Figure 3.4: Infinite model for $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$.

Take into account that when we evaluate a formula with a global operator, such as $\langle \mathsf{gsb} \rangle$, we do not move to an accessible state in which evaluation continues (as is the case for the local operators, such as $\langle \mathsf{sb} \rangle$), instead, we stay in the same state and evaluation continues where we are at. This allow us to change the structure from the current point so that we can later traverse it looking for the changes we made. More precisely, looking at Figure 3.4, we can identify the successors of the spy point by deleting the arrows that are pointing to s-points: from the spy point, we can delete the arrows with the global sabotage operator, and then we can traverse the chain of states looking for the deleted arrows. Note that there is no need to enforce arrows returning to the spy point, we only need to enforce that each successor has an arrow to an s-point. As our intention is to delete those arrows, tough, we need to ensure that such arrows are unique. It's like if each point of the infinite chain had a flag that could be turned on or off to identify a point.

With these observations we can see that formula φ is similar to the one of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, 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. With formula (3) we enforce a serial chain of states; formula (5) makes the evaluation point a "spy point"; formula (6) ensures that each successor is irreflexive; and finally, formulas (7) and (8) together ensure transitivity.

Note that from (2) on each part of φ begins with $\Box\psi$ or $[\mathsf{gsb}]\psi$, meaning that ψ is true at all points of the chain. Hence the spy point can access any state, *describing* and *changing* the model.

3.3 Bridge Logic

We now move on to another kind of relation-changing operator: the one which can create arrows to inaccessible states.

3.3.1 Local Bridge

We exhibit a formula φ, which is a conjunction of several properties:

$$\begin{aligned}
\varphi = \quad & s \wedge [\mathsf{br}]\neg s \wedge \langle \mathsf{br}\rangle \top && (1)\\
& \wedge \Box[\mathsf{br}]\neg s && (2)\\
& \wedge \Box\Box s && (3)\\
& \wedge \Box[\mathsf{br}]\Box\neg s && (4)\\
& \wedge [\mathsf{br}]\Diamond\top && (5)\\
& \wedge [\mathsf{br}][\mathsf{br}](s \to \Box(\neg s \to \Box(\neg s \to \Box\neg s))) && (6)\\
& \wedge [\mathsf{br}]\Box\Box[\mathsf{br}](s \to \Diamond(\neg s \wedge \Diamond\Diamond s)) && (7)
\end{aligned}$$

The intended model is illustrated in Figure 3.5, where the dotted lines represent the arrows created with the $\langle \mathsf{br}\rangle$ operator.

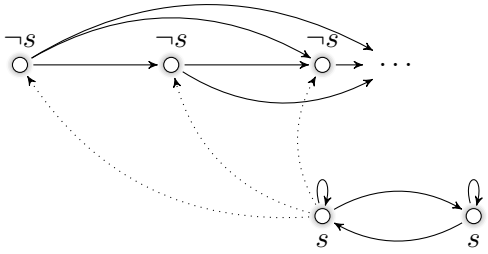


Figure 3.5: Infinite model for $\mathcal{ML}(\langle \mathsf{br}\rangle)$.

Intuitively, the idea is that we basically have two set of points: the set of points in which s is true, and the set of points in which $\neg s$ is true. The first set (located at the bottom of Figure 3.5) forms a connected component, and even though there may be several s-points connected among them, we force only one of those points to be *the* spy point. The second set (located at the top of Figure 3.5) is the partial order set the spy point is going to talk about: a set of *inaccessible* states. Formulas (1) to (5) are mainly devoted to enforce these kind of sets, in addition of creating and infinite chain of states. We choose to define such a model because $\mathcal{ML}(\langle \mathsf{br}\rangle)$ can create edges: as we have inaccessible points, we can create arrows to access those points. On the other hand, because we cannot create an arrow with the $\langle \mathsf{br}\rangle$ operator if it is already present, we force the existence of a set of s-points that are all connected, so that the only arrows we can create go to $\neg s$-states. In this way we can enforce an infinite chain of states that is irreflexive (formula (6)) and transitive (formula (7)).

Let us now see with some level of detail how each part of φ works to get the intended model. Formula (1) establishes that s is true at the evaluation point and that $\neg s$ is true at any inaccessible state from there (in particular, it says that the

evaluation point is reflexive). The last part of (1), namely $\langle \mathsf{br} \rangle \top$, ensures that there is at least one inaccessible state, making the set of states the spy point is going to talk about not empty.

Formulas (2), (3), and (4) are of the form $\Box\psi$, meaning that ψ is true at all successors: if present, these states cannot be states where $\neg s$ is true because they are inaccessible from the evaluation point, so formulas (2), (3), and (4) talk about the set of s-states. In particular, formula (2), where $\psi = [\mathsf{br}]\neg s$, ensures that there are no unconnected states, making the set of points satisfying s a connected component. Formula (3), where $\psi = \Box s$, which can also be rewritten as $\psi = \neg\Diamond\neg s$, says that every s-state of the set has inaccessible $\neg s$-points, while formula (4), where $\psi = [\mathsf{br}]\Box\neg s$, which can also be rewritten as $\psi = [\mathsf{br}]\neg\Diamond s$, says that there are no arrows from inaccessible states incoming into the connected component satisfying s. Observe that, in Figure 3.5, we have only drawn two points from the set of s-states, one of them (the left one) being the spy point. But this set might have one, two, three or more points, all of them connected between each other.

Formulas (5), (6), and (7) are of the form $[\mathsf{br}]\psi$, meaning that ψ is true at all states that are inaccessible from the evaluation point, so formulas (5), (6), and (7) talk about the set of $\neg s$-states. In particular, Formula (5) creates an infinite chain of inaccessible states, while formula (6) ensures that those states are irreflexive: after creating arrows back and forth from the spy point to any inaccessible state (identifying in this way the point we are visiting), all the successors of the visited state cannot see a state that satisfies s, i.e., cannot see the spy point. If the state were reflexive, after doing one loop it would be possible to reach the spy point, because we created the arrow to access it. Hence the visited state has to be irreflexive. Finally, formula (7) ensures transitivity: the idea is that we can create an arrow to an inaccessible state, call it w_1, we can move two steps to reach another state, call it w_3, and from there we can create a new arrow back to the evaluation point. Then we force the existence of an arrow from w_1 to w_3 by traversing the created arrows.

From formulas (6) and (7) it should be clear how we identify the spy point from the other s-points: from the evaluation point, we use the bridge operator to create an arrow to an inaccessible state, i.e., to a state in which $\neg s$ is true, and from there or another $\neg s$-point, we use again the bridge operator to create an arrow to an s-point, which should satisfy a formula of the form $\Diamond\neg s$. As we had already created an arrow to a $\neg s$-state from the spy point, this is the only s-point that can satisfy the latter formula.

Proposition 3.3.1. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model and $s \in W$ a point of the model. If $\mathcal{M}, s \models \varphi$ then $\mathcal{M}$ is infinite.*

Proof. Suppose $\mathcal{M}, s \models \varphi$. Let $B = \{w \in W \mid (s, w) \notin R\}$. Because (1) is satisfied, $s \notin B$ and $B \neq \emptyset$. As (2), (3), and (4) are satisfied, we have $(s, s) \in R$, $(s, w) \notin R$, and $(w, s) \notin R$, respectively, for $w \in W$. These three formulas ensure that s is reflexive, and that the points of B cannot access or be accessed by s. As (5) is satisfied, every point in B has an R-successor, which by (3) and (4) is isolated from s. As (6) is satisfied, every point in B is irreflexive; and because (7) is satisfied, R transitively orders B. Hence B is an unbounded strict partial order, thus B is infinite and so is $\mathcal{M}$. □

Proposition 3.3.2. *There is an infinite model $\mathcal{M}$ and a point s such that $\mathcal{M}, s \models \varphi$.*

Proof. Let $(\mathbb{N}, <)$ be the natural numbers in their usual order, and suppose $s \notin \mathbb{N}$. Let $\mathcal{M} = \langle W, R, V\rangle$ be a model where W is $\mathbb{N} \cup \{s\}$, R is $< \cup \{(s, s)\}$, and V is any

valuation such that $V(s) = \{s\}$ and $V(n) \cap V(s) = \emptyset$, for $n \in \mathbb{N}$. It is clear that $\mathcal{M}, s \models \varphi$. Thus φ has at least one infinite model. □

3.3.2 Global Bridge

We exhibit a formula φ, which is a conjunction of several properties:

$$
\begin{aligned}
\varphi = \quad & s \wedge \Box\neg s \wedge \Diamond\top && (1)\\
& \wedge \Box\Box\neg s && (2)\\
& \wedge \Box\Diamond\neg s && (3)\\
& \wedge \Box\Box(\neg s \to \Box\neg s) && (4)\\
& \wedge [\mathsf{gbr}](\Diamond(\neg\Diamond s \wedge \Diamond\Diamond s) \;\to\; \Diamond\Diamond s) && (5)\\
& \wedge [\mathsf{gbr}]\Box(\Diamond s \to \Box\neg\Diamond s) && (6)\\
& \wedge [\mathsf{gbr}]\neg\Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \Diamond(\neg s \wedge \neg\Diamond s \wedge \Diamond\Diamond s))) && (7)\\
& \wedge [\mathsf{gbr}][\mathsf{gbr}](\Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \Diamond(\neg s \wedge \Diamond s))) \;\to\; \Diamond(\Diamond s \wedge \Diamond\Diamond s)) && (8)
\end{aligned}
$$

We enforce an infinite model in a different way to that of $\mathcal{ML}(\langle\mathsf{br}\rangle)$, but in a similar way to that of $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$, as illustrated in Figure 3.6.

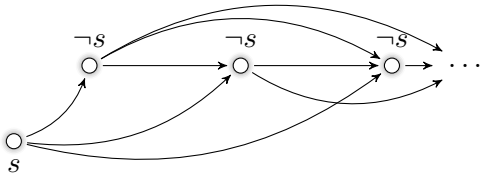


Figure 3.6: Infinite model for $\mathcal{ML}(\langle\mathsf{gbr}\rangle)$.

With φ as defined above we model the exact opposite situation of the global sabotage operator. In the model of $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$ of Figure 3.4, we have arrows pointing to s-points for each state reachable from the evaluation point. We did that because we identified the successor points by deleting those arrows. Now, with the $\langle\mathsf{gbr}\rangle$ operator we can model the same situation but creating arrows instead of deleting them. So we enforce that arrows pointing to s-states cannot exist: in this way we can create arrows to those points to identify the successor states.

Formula φ is similar to the one of $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$, except that (2), which can also be rewritten as $\Box\neg\Diamond s$, establishes that in one step, there are no s-successors, and (4) is the same as (2) but in two steps. The rest of the formulas of $\mathcal{ML}(\langle\mathsf{gbr}\rangle)$ are analogous to the formulas of $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$. In fact, except for a small detail in formula (8), we can obtain formula φ of $\mathcal{ML}(\langle\mathsf{gbr}\rangle)$ from formula φ of $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$ by replacing every occurrence of $[\mathsf{gbr}]$ for $[\mathsf{gsb}]$, every occurrence of $\neg\Diamond s$ for $\Diamond s$, and every occurrence of $\Diamond s$ for $\neg\Diamond s$.

3.4 Swap Logic

We now enforce infinite models for the last kind of relation-changing operator: the swap operator. Define $\Box^0\varphi$ as φ, $\Box^{n+1}\varphi$ as $\Box\Box^n\varphi$, and let $\Box^{(n)}\varphi$ be a shorthand for $\bigwedge_{1\leq i\leq n}\Box^i\varphi$.

We exhibit a formula φ taken from [AFH14], which is a conjunction of several

properties:

$$\begin{aligned}\varphi = \quad & s \wedge \Box^{(9)}\neg s \wedge \Diamond\top \wedge \Box\Diamond\top && (1)\\ & \wedge\ [\mathsf{sw}][\mathsf{sw}](\neg s \to \Diamond\Diamond\Diamond\Diamond s) && (2)\\ & \wedge\ [\mathsf{sw}]\Box\Box\neg s && (3)\\ & \wedge\ [\mathsf{sw}][\mathsf{sw}][\mathsf{sw}](\neg\Diamond s \to \Diamond\Diamond\Diamond(\neg s \wedge \Diamond\Diamond\Diamond s)) && (4)\end{aligned}$$

We exhibit another formula φ', which is a variant of the previous one, adapted for the global operator:

$$\begin{aligned}\varphi' = \quad & s \wedge \Box^{(9)}\neg s \wedge \Diamond\top \wedge \Box\Diamond\top && (1)\\ & \wedge\ \Box\Box[\mathsf{gsw}][\mathsf{gsw}]\Box\Box(s \to \Diamond\Diamond\Diamond s) && (2)\\ & \wedge\ \Box[\mathsf{gsw}](\Diamond s \to \Box\Box\neg s) && (3)\\ & \wedge\ \Box\Box\Box[\mathsf{gsw}][\mathsf{gsw}][\mathsf{gsw}](\Diamond\Diamond\Diamond s \to \Diamond\Diamond\Diamond(\neg s \wedge \Diamond\Diamond\Diamond s)) && (4)\end{aligned}$$

The intended model is illustrated in Figure 3.7.

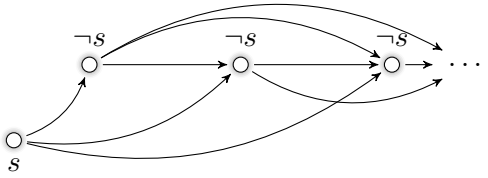


Figure 3.7: Infinite model for $\mathcal{ML}(\langle\mathsf{sw}\rangle)$ and $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$.

Intuitively, the idea is that we can enforce a infinite chain of states by ensuring that the propositional symbol s is true only at the evaluation point, and false in all other states reachable from there. The spy point sees all the points of the model, but the rest of the points cannot see it. Then, φ enforces specific properties on the model, locating states by their distance to the spy point using formulas of the form $\Diamond \dots \Diamond s$, after swapping an outgoing edge from the spy point. 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.

Let us now see with some level of detail how each part of φ works to get the intended model. The first half of (1) makes s true at the evaluation point and false at all states accessible within 9 steps. The second half of (1) ensures that there is at least one successor (making the set of states the spy point is going to talk about not empty), and that each successor reaches some successor.

Formula (2) tells that from any state reachable in two swapping steps, it is possible to go back to the evaluation point in five steps. But this is only possible by first going to the evaluation point in two steps, then going to the visited point in one step, and finally going again to the evaluation point in two steps. Hence all states accessible in two steps from the evaluation point are also accessible in one. This makes the evaluation point a "spy point," i.e., it is directly connected to every state in the submodel generated from it.

Formula (3) enforces irreflexivity: after swapping any outgoing edge from the spy point and arriving some reachable state (identifying in this way the point we are visiting), all the successors of the visited state can only see states that do not satisfy s. If the state were reflexive, after doing one loop it would be possible to reach a state that satisfies s (the spy point), because we arrived to this state from the spy point by swapping an arrow. Hence the visited state has to be irreflexive. Finally, formula (4) enforces transitivity: it does so by enforcing the same property as (3) but now on the successors of the spy point.

Formula φ enforces seriality, irreflexivity, and transitivity on a chain of states for the local variant of swap logic. Formula φ' does the same as formula φ, the only difference in φ' is the use of the global operator, which in turn mimics the local operator. For local swap we first turn around edges with formulas of the form $\langle \mathsf{sw} \rangle \ldots \langle \mathsf{sw} \rangle \psi$ and then we return to the point where we started with an equal number of modalities of the form $\Diamond \ldots \Diamond \psi$. For global swap we can mimic this effect by doing it in the opposite direction, i.e, we first move forward to a state with formulas of the form $\Diamond \ldots \Diamond \psi$ and then we swap around edges, from the point where we are at, with an equal number of modalities of the form $\langle \mathsf{gsw} \rangle \ldots \langle \mathsf{gsw} \rangle \psi$. As we have turned around the same amount of edges that we have traversed we can enforce formulas that let us go back to the point where we started. In this way, the global operator mimics the formulas of the local operator.

Proposition 3.4.1. *Let $\mathcal{M} = \langle W, R, V \rangle$ be a model and $s \in W$ a point of the model. If $\mathcal{M}, s \models \varphi$ then $\mathcal{M}$ is infinite.*

Proof. Suppose $\mathcal{M}, s \models \varphi$. Let $B = \{w \in W \mid (s, w) \in R\}$. Because (1) is satisfied, $s \notin B$, $B \neq \emptyset$, and every point in B has an R-successor distinct from s. Because (2) is satisfied, if $w \neq s$ and w is an R-successor of an element of B then w is also an element of B. As (3) is satisfied, every point in B is irreflexive; and because (4) is satisfied, R transitively orders B. Hence B is an unbounded strict partial order, thus B is infinite and so is $\mathcal{M}$. □

Proposition 3.4.2. *There is an infinite model $\mathcal{M}$ and a point s such that $\mathcal{M}, s \models \varphi$.*

Proof. Let $(\mathbb{N}, <)$ be the natural numbers in their usual order, and suppose $s \notin \mathbb{N}$. Let $\mathcal{M} = \langle W, R, V \rangle$ be a model where W is $\mathbb{N} \cup \{s\}$, R is $< \cup \{(s, n) \mid n \in \mathbb{N}\}$, and V is any valuation such that $V(s) = \{s\}$ and $V(n) \cap V(s) = \emptyset$, for $n \in \mathbb{N}$. It is clear that $\mathcal{M}, s \models \varphi$. Thus φ has at least one infinite model. □

To sum up, in this chapter we showed that relation-changing logics lack the finite model property by presenting formulas that can only be satisfied in infinite models. The idea was to use a *spy point technique*. A spy point, as we saw in this chapter, is a unique state from which we can *describe* and *change* the model; we used different kind of spy points according to the expressivity of each logic. For $\langle \mathsf{sb} \rangle$ we used a spy point that is connected back and forth with every point of the model; for $\langle \mathsf{br} \rangle$ we defined a spy point that is disconnected from every point of the infinite chain; and for $\langle \mathsf{sw} \rangle$ we used a spy point that can reach (but cannot be reached by) all the points of the model. Defining spy points with the global versions of sabotage, bridge, and swap was more easy because the global operators can change edges anywhere in the model, and therefore are more expressive. For $\langle \mathsf{gsb} \rangle$ we used a spy point that can see (but cannot be seen by) all the points of the model, and additionally, we forced the *existence* of a particular arrow in each point of the infinite chain so that we can later *delete* it in order to identify the states; for $\langle \mathsf{gbr} \rangle$ we defined a spy point in a similar way to that of $\langle \mathsf{gsb} \rangle$, but modeling the opposite situation: we forced the *non-existence* of a particular arrow in each point of the infinite chain so that we can later *create* it in order to identify the states; and for $\langle \mathsf{gsw} \rangle$ we used the same spy point as for the local swap operator aiming at mimicking its behavior. After defining the spy points, we wrote formulas to enforce serial, irreflexive, and transitive models, implying that the models are infinite.

As we discussed at the beginning of this chapter, the finite model property is a key property to have decidability. The ability to enforce infinite models shows that

the logics are very expressive, and we will see that it is due to this high expressive power that they can cross the border of decidability. We will prove undecidability for relation-changing logics in the next chapter.

CHAPTER 4

Undecidability Results

4.1 The Satisfiability Problem

A classical problem that is interesting to investigate when we study logic is the *satisfiability problem*: given a formula φ, is φ satisfiable in some model? Informally, a logic is said to be decidable if the satisfiability problem is decidable. That is, it is possible (ignoring constraints of time and space) to write a computer program which takes a formula as input, and halts after a finite number of steps and correctly tells us whether it is satisfiable in some model or not. If we cannot write such a computer program to decide the satisfiability problem, the logic is said to be undecidable.

In this chapter we will prove that the sabotage, bridge, and swap logics, both with local and glocal effects, are undecidable. The question then is: how do we prove undecidability? Given a modal satisfiability problem S, to prove that S in undecidable we must reduce some known undecidable problem U to S. But which problems are the interesting candidates for reduction? Unsurprisingly, there is no single best answer to this question. There are many candidates for making the reduction, but certain problems are particularly suitable to modal logic: tiling problems are a nice example [Wan61, BGG01].

4.1.1 Undecidability via Tiling

A tiling problem is in essence a jigsaw puzzle. A tile t is simply a 1×1 square, fixed in orientation, each side of which has a color; we refer to these four colors as $right(t)$, $left(t)$, $up(t)$, and $down(t)$. The general form that tiling problems take is: given a finite set $\mathcal{T}$ of distinct types of tile, can we cover a certain part of $\mathbb{Z} \times \mathbb{Z}$ in such a way that adjacent tiles have the same color on the neighboring sides? Covering a grid with tiles so that adjacent colors match is called *tiling*. This simple idea of patter matching underling tiling problems gives rise to a family of problems which can be used to demonstrate undecidability and complexity results (see [Har85, Har86] for a demostration of the flexibility of the method as a tool for measuring the complexity of logics). In particular, the $\mathbb{N} \times \mathbb{N}$ tiling problem is: given a finite set of tile types $\mathcal{T}$, can $\mathcal{T}$ tile $\mathbb{N} \times \mathbb{N}$? This problem is hard, and in fact it is known to be undecidable (proofs that the $\mathbb{N} \times \mathbb{N}$ tiling problem is undecidable can be found in [Ber66, Rob71, LP97]).

We first attempted to prove undecidibility for relation-changing logics via a

reduction of the $\mathbb{N} \times \mathbb{N}$ tiling problem, obtaining the desired results: the logics turned out to be undecidable. We outline this proof in the following theorem.

Theorem 4.1.1. *The satisfiability problem of* $\mathcal{ML}(\blacklozenge)$ *is undecidable, for* $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle\}$.

Proof (Sketch). We proceed by reducing the $\mathbb{N} \times \mathbb{N}$ tiling problem to the $\mathcal{ML}(\blacklozenge)$ satisfiability problem. Let $\mathcal{T} = \{T_1, \ldots, T_k\}$ be the given set of tile types. The idea is to construct a formula $\varphi^{\mathcal{T}}$ such that $\mathcal{T}$ tiles $\mathbb{N} \times \mathbb{N}$ if and only if $\varphi^{\mathcal{T}}$ is satisfiable.

To encode the $\mathbb{N} \times \mathbb{N}$ tiling problem into the $\mathcal{ML}(\blacklozenge)$ satisfiability problem we use three modalities: $\langle \mathsf{s} \rangle$, $\langle \mathsf{u} \rangle$, and $\langle \mathsf{r} \rangle$. We use the $\langle \mathsf{s} \rangle$ modality to define a spy point (i.e, the point of evaluation have access in one $\langle \mathsf{s} \rangle$-step to any reachable state in the model), and we use the $\langle \mathsf{u} \rangle$ and $\langle \mathsf{r} \rangle$ modalities to represent movement up and to the right, respectively, from one tile to the other; we also encode each type of tile with a fixed propositional symbol t_i. With this encoding the construction of $\varphi^{\mathcal{T}}$ proceeds in three steps. First, we use each $\mathcal{ML}(\blacklozenge)$-ability of changing the model to define the spy point and to demand grid-like models (this is the hardest part of the proof). Second, we show how to use each $\mathcal{ML}(\blacklozenge)$ to demand that a tiling exists on this grid. Finally, we show that adjacent tiles have the same color on the common side.

As we succeeded in constructing a formula $\varphi^{\mathcal{T}}$ such that $\mathcal{T}$ tiles $\mathbb{N} \times \mathbb{N}$ if and only if $\varphi^{\mathcal{T}}$ is satisfiable, it follows that the $\mathcal{ML}(\blacklozenge)$ satisfiability problem is undecidable. □

Proving undecidability is something of an art: it can be very difficult, and there is no substitute for genuine insight into the satisfiability problem. The $\mathbb{N} \times \mathbb{N}$ tiling problem is an interesting candidate for reduction that was useful to us for proving undecidability of relation-changing logics. This first attempt was very successful and gave us some insight on the behavior and high expressivity of the logics, but we are not going to show the full proof in this thesis. Instead, we are going to present the proofs of undecidability by reducing a different undecidable problem: the satisfiability problem of *memory logics*. Before explaining why this approach gave us better results, let us first meet memory logics.

4.1.2 Memory Logics

Memory logics are modal logics extended with the ability to *store* the current state of evaluation into a set (the memory) and to *check* whether the current state of evaluation belongs to this set or not. The original idea was introduced in [Are07] and was further studied in Mera's PhD thesis [Mer09]. There are several memory logics, we will just introduce the simplest one formally.

Definition 4.1.2 (Syntax of Memory Logic)**.** Let PROP be a countable, infinite set of propositional symbols. Then the set FORM of formulas over PROP is defined as:

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

where $p \in \mathsf{PROP}$ and $\varphi, \psi \in \mathsf{FORM}$. Other operators are defined as usual.

We call $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ the extension of $\mathcal{ML}$ allowing also the $\textcircled{r}$ and $\textcircled{k}$ operators, which stand for "remember" and "known," respectively.

We now move to the formal semantics. A model $\mathcal{M} = \langle W, R, V, M \rangle$ is an extension of a Kripke model with an extra set $M \subseteq W$. Models of memory logics are Kripke models but with a memory M in which we can store the current state of evaluation.

Definition 4.1.3 (Semantics of Memory Logic)**.** Let w be a state of the model, we inductively define the notion of satisfiability of a formula as:

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

A formula φ of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ is *satisfiable* if there is a model $\langle W, R, V, \emptyset\rangle$ such that $\langle W, R, V, \emptyset\rangle, w \models \varphi$. The empty initial memory ensures that no point of the model satisfies the unary predicate $\textcircled{k}$ unless a formula $\textcircled{r}\varphi$ has previously been evaluated there.

Note that when we evaluate $\textcircled{r}\varphi$, the remember operator *changes* the model (a new state is added to the memory), and φ is evaluated in the modified model. The $\textcircled{k}$ operator simply checks if the current state of evaluation is in the memory or not. Thus $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ allow us to model dynamic behavior through an explicit memory operator that changes the evaluating structure.

While relation-changing logics allow us to change the model through operators that *change the accessibility relation*, memory logics allow us to change the model through operators that *change the memory*. Note the analogy between these two families of dynamic logics: both are extensions of the basic modal logic and increase its expressive power [Fer14a, Mer09]. In particular, the model checking problem of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ has been proven to be PSPACE-complete [AFGM09, Mer09], i.e., it is decidable and has the same complexity as $\mathcal{ML}(\blacklozenge)$, for $\blacklozenge \in \{\langle \mathsf{sb}\rangle, \langle \mathsf{gsb}\rangle, \langle \mathsf{br}\rangle, \langle \mathsf{gbr}\rangle, \langle \mathsf{sw}\rangle, \langle \mathsf{gsw}\rangle\}$. On the other hand, the satisfiability problem of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ has been shown to be undecidable via a reduction of the $\mathbb{N} \times \mathbb{N}$ tiling problem [AFFM11, AFFM08] in a similar way as we did for the proof of Theorem 4.1.1. It seems natural, then, that the satisfiability problems of sabotage, bridge, and swap logics are also undecidable. That is what we will prove in this chapter, but before moving on let us first see an example to explore further the relationship between these two kind of dynamic logics.

Consider model $\mathcal{M} = \langle W, R, V, \emptyset\rangle$ of Figure 4.1, where $W = \{w_1, w_2\}$, $R = \{(w_1, w_2), (w_2, w_1), (w_2, w_2)\}$, and where there are no propositional symbols.

Figure 4.1: Example model.

At w_1 we can evaluate the formula $\Diamond\Diamond\top$. That is, we can go to a successor of w_1, which is w_2, and from there we have two options: we can return to w_1 or we can stay in w_2. How can we distinguish that after reaching w_2 we can go to a successor different from itself? We can use memory operators to distinguish this situation. We can check whether the formula

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

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

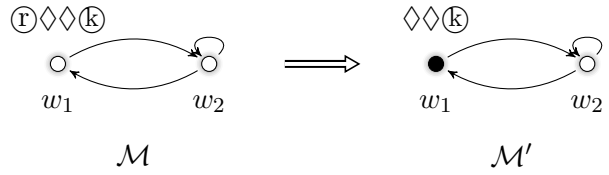


Figure 4.2: Changing the model with $\mathcal{ML}(ⓡ, ⓚ)$.

So now the remaining formula ◊◊ⓚ is evaluated in $\mathcal{M}'$. To be satisfiable, this formula needs a w_1-successor, which is w_2, and from there needs another succesor w_i such that $w_i \in M$, for $i = 1, 2$. Because w_1 is in the memory in $\mathcal{M}'$, that successor w_i must be w_1. Observe that if we had not memorized w_1, after reaching w_2 we would not be sure if the next successor is in fact different from w_2. Storing points in the memory is equivalent to labeling a node as "visited" so that later on we can check if the current point of evaluation has been visited before.

So far, so good. Now we want to show that we can express the same situation in a standard Kripke model, without an additional memory. How? Changing the accessibility relation instead of changing the memory. Consider model $\mathcal{M} = \langle W, R, V\rangle$ as defined before but without the set M. In the model of Figure 4.1, we can simulate the formula ⓡ◊◊ⓚ of $\mathcal{ML}(ⓡ, ⓚ)$ with a formula of $\mathcal{ML}(\langle \mathsf{sb}\rangle)$. We can check whether the formula

$$\langle \mathsf{sb}\rangle \Diamond \neg \Diamond \top$$

is satisfiable at w_1. As you can see, the $\langle \mathsf{sb}\rangle$ operator is used to go to a successor while we delete the traversed arrow. Deleting the adjacent edge of w_1 can be seen as equivalent as remembering w_1. After evaluating the sabotage operator the previous model turns into $\mathcal{M}' = \langle W, R\backslash\{(w_1, w_2)\}, V\rangle$:

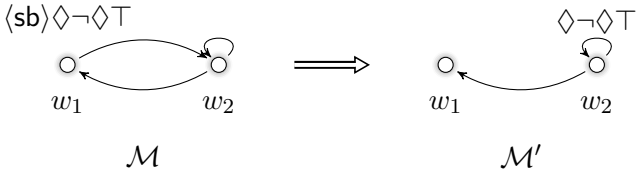


Figure 4.3: Changing the model with $\mathcal{ML}(\langle \mathsf{sb}\rangle)$.

So now the remaining formula $\Diamond\neg\Diamond\top$ is evaluated at w_2 in $\mathcal{M}'$. To be satisfiable, this formula needs a successor such that $\neg\Diamond\top$ holds, i.e., a successor that does not have a successor. As we deleted the successor of w_1, the formula is satisfiable only at w_1. Observe that if we had not deleted the successor of w_1, after reaching w_2 we would not be sure if the next successor is in fact different from w_2. Deleting the successor of w_1 can be seen as storing w_1 into a memory, and checking for the deleted arrow can be seen as "knowing" if w_1 is indeed in the memory.

In a similar way, we can define memory logic models in which we can simulate the ⓡ and ⓚ operators with standard Kripke models using operators that can swap and add edges, so that later on we can check for the edges that have been swapped or added. Hence, the undecidable satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$ seems a good candidate for making the reductions in the undecidability proofs.

Now that we have presented memory logics, let us go back to the discussion of undecidability. To prove that relation-changing logics are undecidable, we attacked the problem in two different ways: by encoding the $\mathbb{N} \times \mathbb{N}$ tiling problem, and by encoding the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$. The $\mathbb{N} \times \mathbb{N}$ tiling problem was our

first attempt but the reduction is not as intuitive as the reduction of the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$. Moreover, in the proof of Theorem 4.1.1, we used three relations R_s, R_u, and R_r: R_s for moving from the spy point to every other point in the model, and R_u, R_r for moving up and right, respectively, from one tile to the other. With memory logics, on the other hand, we will see that only one relation R is enough. Besides, we will also see that the spy points defined for the infinite models of Chapter 3 play a crucial role in the encoding of the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$: we can reuse and adapt the work of our previous chapter.

For these reasons, we choose to present in this thesis the proofs of undecidability via memory logics. We will provide reductions for each of our six logics. As usual, we start with sabotage logic.

4.2 Sabotage Logic

4.2.1 Local Sabotage

We define a computable function $\tau : \mathsf{FORM}_{\mathcal{ML}(ⓡ,ⓚ)} \to \mathsf{FORM}_{\mathcal{ML}(\langle \mathsf{sb} \rangle)}$ that reduces the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$ to the satisfiability problem of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$. That is, for each formula φ satisfiable in some $\mathcal{ML}(ⓡ, ⓚ)$-model we construct a formula $\tau(\varphi)$ satisfiable in some $\mathcal{ML}(\langle \mathsf{sb} \rangle)$-model such that:

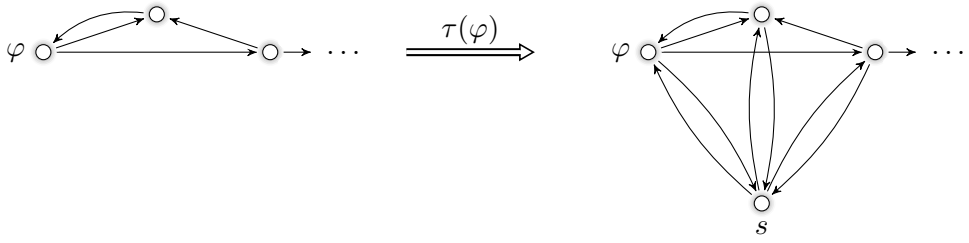
$$\varphi \text{ is satisfiable if and only if } \tau(\varphi) \text{ is satisfiable.} \tag{4.1}$$

The undecidability of the satisfiability problem of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ will follow from the undecidability of the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$.

The construction of τ proceeds in two steps. First, we translate memory logic models to standard Kripke models. We enforce some constraints on the structure of our translated models with a formula *Struct* that enlarges the original $\mathcal{ML}(ⓡ, ⓚ)$-model with a spy point: we enforce arrows departing from and returning to the spy point as we did for the infinite models of Section 3.2.1. And finally, in order for a Kripke model to simulate a memory, we provide a translation Tr from $\mathcal{ML}(ⓡ, ⓚ)$-formulas to $\mathcal{ML}(\langle \mathsf{sb} \rangle)$-formulas that simulates the ⓡ and ⓚ operators: storing a point in the memory is simulated by deleting some of the arrows we have enforced in the spy point, and checking whether the current point of evaluation is in the memory is simulated by checking if an arrow has been deleted or not.

Definition 4.2.1. Let φ be an $\mathcal{ML}(ⓡ, ⓚ)$-formula that does not contain the propositional symbol s. We define $\tau(\varphi) = \mathit{Struct} \wedge \mathsf{Tr}(\varphi)$.

The effect of $\tau(\varphi)$ for some φ is illustrated below:



If we succeed in constructing such a function τ, it follows that the satisfiability problem of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ is undecidable. Why? Well, suppose it were decidable. Then we could solve the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$ as follows: given φ, we build $\tau(\varphi)$ and use the decision procedure of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ to decide whether $\tau(\varphi)$ is

satisfiable. By (4.1) this would solve the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$, which is impossible given that it is known to be undecidable.

Let us now see how we can define *Struct* and $\mathsf{Tr}(\varphi)$ to make the reduction.

Definition 4.2.2. Let $\mathcal{M} = \langle W, R, V, M\rangle$ be an $\mathcal{ML}(ⓡ, ⓚ)$-model in which φ is satisfied. We define *Struct* as the conjunction of several properties:

$$
\begin{array}{lll}
\mathit{Struct} = & s \wedge \Box\neg s \wedge \Box\Diamond s & (1') \\
& \wedge\ \Box\Box(s \to \Box\neg s) & (2') \\
& \wedge\ [\mathsf{sb}][\mathsf{sb}](s \to \Box\Diamond s) & (3') \\
& \wedge\ \Box[\mathsf{sb}](s \to \Diamond\neg\Diamond s) & (4') \\
& \wedge\ \Box\Box(\neg s \to \Diamond(s \wedge \neg\Diamond s)) & (5') \\
& \wedge\ \Box[\mathsf{sb}](\neg s \to [\mathsf{sb}](s \to \Box\Box(\neg s \to \Diamond s))) & (6') \\
& \wedge\ \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\Diamond(\neg s \wedge \neg\Diamond s))) & (7') \\
& \wedge\ \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\neg\Diamond s)) & (8')
\end{array}
$$

Struct defines the structure of our translated models by enlarging the original model with a spy point. Note that formulas $(1')$ to $(8')$ are analogous to formulas of the infinite models of Section 3.2.1; the main difference is that we don't need to enforce a non-empty set of points ($\Diamond\top$) in $(1')$, and in a similar way we don't need to enforce seriality, irreflexivity, and transitivity. By dropping these properties, the rest of the formulas remain the same: formulas $(2')$, $(3')$, and $(4')$ enforce arrows back to the evaluation point in one step, while formulas $(5')$, $(6')$, and $(7')$ do the same but in two steps. $(8')$ ensures that the evaluation point is linked to every point of the model except to itself, i.e., makes the evaluation point a "spy point."

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

Proposition 4.2.3. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model without a memory and $w \in W$. If $\mathcal{M}, w \models$ Struct, then the following properties hold:*

1. *w is the only state of the model that satisfies the propositional symbol s, in the connected component generated by w.*
2. *For all states $v \in W$ such that $v \neq w$, if $(w, v) \in R$ then $(v, w) \in R$, and w is a spy point, i.e., if $(w, v) \in R^*$ then $(w, v) \in R$ – where R^* denotes the reflexive-transitive closure of R.*

Proposition 4.2.3 enumerates the main properties of the spy point: it is the only point in the connected component satisfying s, 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 of $\mathcal{ML}(ⓡ, ⓚ)$-formulas to $\mathcal{ML}(\langle\mathsf{sb}\rangle)$-formulas.

Definition 4.2.4. Given φ, we define $\mathsf{Tr}(\varphi) = \Diamond(\varphi)'$, where $(\ )'$ is defined as:

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

$\mathsf{Tr}(\varphi)$ places the translation $(\ )'$ of the memory logic formula φ right after the evaluation point. As you can see, $(\ )'$ is defined inductively on the structure of the

formulas. Boolean cases are obvious. For the diamond case, $\Diamond\psi$ is satisfied if there is a successor v where ψ holds, but we must ensure that v is a point of the original $\mathcal{ML}(ⓡ, ⓚ)$-model and not the spy point, so v cannot satisfy s. ⓡ is represented by removing the edge from the state we want to memorize to the spy point, and from the spy point to the "memorized" state. Observe how the translation behaves: if the point has already been memorized ($\neg\Diamond s$), then nothing needs to be done and the translation continues; otherwise ($\Diamond s$), we make the spy point inaccessible using $\langle \mathsf{sb} \rangle$, and we also delete the arrow from the spy point to point that cannot access it. ⓚ is represented by checking whether there is an edge pointing to the spy point: if such an edge does not exist it means that the current point of evaluation is in the memory.

Theorem 4.2.5. *Let φ be a formula of $\mathcal{ML}(ⓡ, ⓚ)$ that does not contain the propositional symbol s. Then, φ is satisfiable if and only if $\tau(\varphi)$ is satisfiable.*

Proof. **(φ is sat $\Leftarrow$ $\tau(\varphi)$ is sat)** Suppose that $\tau(\varphi)$ is satisfiable, i.e., that there exists a model $\mathcal{M} = \langle W, R, V\rangle$ and $s \in W$ such that $\langle W, R, V\rangle, s \models \mathit{Struct}$ and $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$. We want to show that φ is satisfiable, i.e., that there is a model $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$ and $w' \in W'$ such that $\langle W', R', V', \emptyset\rangle, w' \models \varphi$. As $\langle W, R, V\rangle, s \models \mathit{Struct}$ we can define $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$, where

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

Thus $\mathcal{M}'$ is extracted from the translated model $\mathcal{M}$, and because $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$, there is $w' \in W'$ such that $(s, w') \in R$ and $\langle W, R, V\rangle, w' \models (\varphi)'$.

We can see that the memory in $\mathcal{M}'$ is initially empty, but as soon as we start remembering elements we need to delete pairs of arrows to simulate storing those elements in the memory, so we need to keep track of the changes in the accessibility relation. We will prove:

$$\langle W', R', V', M'\rangle, v' \models \psi \;\text{ iff }\; \langle W, R_{M'}, V\rangle, v' \models (\psi)' \tag{4.2}$$

where $v' \in W'$, $M' \subseteq W'$, $\psi \in \mathsf{FORM}$, and $R_{M'} = R \backslash \{(m', s), (s, m') \mid m' \in M'\}$. In particular, when $M' = \emptyset$ we have that $\langle W', R', V', \emptyset\rangle, w' \models \varphi$ iff $\langle W, R, V\rangle, w' \models (\varphi)'$.

We now prove (4.2) by structural induction on ψ, or more precisely, by structural induction on the number of connectives in ψ.

First suppose that ψ contains no connectives. Then ψ could be $\bot$, a propositional symbol p, or the unary predicate ⓚ. For the purposes of the inductive proofs we regard $\bot$ as a propositional symbol rather than as a logical connective (but observe that $\bot$ is trivially false at v' in both models, so we have the desired equivalence). Hence, we have two base cases:

- $\boldsymbol{\psi = p}$: Suppose that $\langle W', R', V', M'\rangle, v' \models p$. By the semantics 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 the semantics 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 the semantics we have $v' \in M'$, then by definition of $R_{M'}$ and Proposition 4.2.3 we have $\{(v', s), (s, v')\} \not\subseteq R_{M'}$ and $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s$. Then by definition of $(\ )'$ this is equivalent to $\langle W, R_{M'}, V\rangle, v' \models (ⓚ)'$.

Now for the inductive case. The induction hypothesis is that the desired equivalence holds for all formulas containing at most n connectives (where $n \geq 0$):

$$\begin{aligned} &\langle W', R', V', M' \rangle, v' \models \phi \text{ iff } \langle W, R_{M'}, V \rangle, v' \models (\phi)' \\ &\langle W', R', V', M' \rangle, v' \models \chi \text{ iff } \langle W, R_{M'}, V \rangle, v' \models (\chi)' \end{aligned} \tag{I.H}$$

We must now show that the equivalence holds for all formulas ψ containing $n+1$ connectives. Then ψ could be either $\neg\phi$, $\phi \wedge \chi$, $\Diamond\phi$, or $\textcircled{r}\phi$. Hence, we have the following inductive cases:

$\boldsymbol{\psi = \neg\phi}$: Suppose that $\langle W', R', V', M' \rangle, v' \models \neg\phi$. By the semantics we have $\langle W', R', V', M' \rangle, v' \not\models \phi$. By induction hypothesis we have $\langle W, R_{M'}, V \rangle, v' \not\models (\phi)'$, but this is the case if and only if $\langle W, R_{M'}, V \rangle, v' \models \neg(\phi)'$. Then, by definition of $(\ )'$ this is equivalent to $\langle W, R_{M'}, V \rangle, v' \models (\neg\phi)'$.

$\boldsymbol{\psi = \phi \wedge \chi}$: Suppose that $\langle W', R', V', M' \rangle, v' \models \phi \wedge \chi$. By the semantics we have $\langle W', R', V', M' \rangle, v' \models \phi$ and $\langle W', R', V', M \rangle, v' \models \chi$. By induction hypothesis we have $\langle W, R_{M'}, V \rangle, v' \models (\phi)'$ and $\langle W, R_{M'}, V \rangle, v' \models (\chi)'$. Then by the semantics we have $\langle W, R_{M'}, V \rangle, v' \models (\phi)' \wedge (\chi)'$, which by definition of $(\ )'$ is equivalent to $\langle W, R_{M'}, V \rangle, v' \models (\phi \wedge \chi)'$.

$\boldsymbol{\psi = \Diamond\phi}$:

($\Leftarrow$) Suppose that $\langle W, R_{M'}, V \rangle, v' \models (\Diamond\phi)'$. By definition of $(\ )'$ we have $\langle W, R_{M'}, V \rangle, v' \models \Diamond(\neg s \wedge (\phi)')$. By the semantics there is $u' \in W$ such that $(v', u') \in R_{M'}$ and $\langle W, R_{M'}, V \rangle, u' \models \neg s \wedge (\phi)'$. Then by the semantics we have $\langle W, R_{M'}, V \rangle, u' \models \neg s$ and $\langle W, R_{M'}, V \rangle, u' \models (\phi)'$. By applying induction hypothesis on the second conjunct we have $\langle W', R', V', M' \rangle, u' \models \phi$, hence by the semantics we have $\langle W', R', V', M' \rangle, v' \models \Diamond\phi$.

($\Rightarrow$) Suppose that $\langle W', R', V', M' \rangle, v' \models \Diamond\phi$. By the semantics there is $u' \in W'$ such that $(v', u') \in R'$ and $\langle W', R', V', M' \rangle, u' \models \phi$. Then by induction hypothesis we have $\langle W, R_{M'}, V \rangle, u' \models (\phi)'$, and by Proposition 4.2.3 and the semantics we have $\langle W, R_{M'}, V \rangle, u' \models \neg s \wedge (\phi)'$. Hence by the semantics we have $\langle W, R_{M'}, V \rangle, v' \models \Diamond(\neg s \wedge (\phi)')$, which is equivalent to $\langle W, R_{M'}, V \rangle, v' \models (\Diamond\phi)'$ by definition of $(\ )'$.

$\boldsymbol{\psi = \textcircled{r}\phi}$:

($\Leftarrow$) Suppose that $\langle W, R_{M'}, V \rangle, v' \models (\textcircled{r}\phi)'$. By definition of $(\ )'$ and the semantics we have the conjunction of the following formulas:

$$\begin{aligned} &(1) \quad \langle W, R_{M'}, V \rangle, v' \models \Diamond s \rightarrow \langle \mathsf{sb} \rangle (s \wedge \langle \mathsf{sb} \rangle (\neg\Diamond s \wedge (\phi)')) \\ &(2) \quad \langle W, R_{M'}, V \rangle, v' \models \neg\Diamond s \rightarrow (\phi)' \end{aligned}$$

We have to prove that $\langle W', R', V', M' \rangle, v' \models \textcircled{r}\phi$. We will prove it in two parts, first assuming (1) and then assuming (2).

First, assume (1). Now suppose $\langle W, R_{M'}, V \rangle, v' \models \Diamond s$, i.e, that $(v', s) \in R_{M'}$. Then we have 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 the semantics and because we assumed $(v', s) \in R_{M'}$ we have $\langle W, (R_{M'})^{-}_{v's}, V \rangle, s \models s \wedge \langle \mathsf{sb} \rangle (\neg\Diamond s \wedge (\phi)')$. It is easy to see that $\langle W, (R_{M'})^{-}_{v's}, V \rangle, s \models s$, so let us check $\langle W, (R_{M'})^{-}_{v's}, V \rangle, s \models \langle \mathsf{sb} \rangle (\neg\Diamond s \wedge (\phi)')$. By the semantics and because $(s, v') \in (R_{M'})^{-}_{v's}$ by Proposition 4.2.3, we have $\langle W, (R_{M'})^{-}_{v's,sv'}, V \rangle, v' \models \neg\Diamond s \wedge (\phi)'$. The first conjunct is trivial because $(v', s) \notin (R_{M'})^{-}_{v's,sv'}$ and by Proposition 4.2.3, so by applying induction hypothesis on the second conjunct, and because we

know $v' \in M'$ by definition of $R_{M'}$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$. Hence by the semantics we have $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$.

Finally, assume (2). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s$, i.e, that $(v', s) \notin R_{M'}$, which in turn also tell us that $(s, v') \notin R_{M'}$, by definition of $R_{M'}$ and Proposition 4.2.3, and therefore v' is in the memory. Then we have that $\langle W, R_{M'}, V\rangle, v' \models (\phi)'$. By induction hypothesis and because we know $v' \in M'$, we have that $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$, which is equivalent to $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$ by the semantics.

($\Rightarrow$) Suppose $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$. We have to prove $\langle W, R_{M'}, V\rangle, v' \models (ⓡ\phi)'$, i.e, we have to prove $\langle W, R_{M'}, V\rangle, v' \models \Diamond s \to \langle \mathsf{sb}\rangle(s \wedge \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\phi)'))$ and $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s \to (\phi)'$. It is easy to prove each part separately (steps are similar to that of the ($\Leftarrow$) direction).

(φ is sat $\Rightarrow$ $\tau(\varphi)$ is sat) Suppose that φ is satisfiable, i.e., that there exists a model $\mathcal{M} = \langle W, R, V, \emptyset\rangle$ and $w \in W$ such that $\langle W, R, V, \emptyset\rangle, w \models \varphi$. We want to show that $\tau(\varphi)$ is satisfiable, i.e., that there is a model $\mathcal{M}' = \langle W', R', V'\rangle$ and $s \in W'$ such that $\langle W', R', V'\rangle, s \models \mathit{Struct} \wedge \mathsf{Tr}(\varphi)$. 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:

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

By construction of $\mathcal{M}'$ it is easy to check that $\langle W', R', V'\rangle, s \models \mathit{Struct}$, so it only remains to see that $\langle W', R', V'\rangle, s \models \mathsf{Tr}(\varphi)$. We can verify that

$$\langle W, R, V, M\rangle, w \models \varphi \;\text{ iff }\; \langle W', R'_M, V'\rangle, s \models \mathsf{Tr}(\varphi) \tag{4.3}$$

where $R'_M = R'\backslash\{(m, s), (s, m) \mid m \in M\}$.

We prove (4.3) by structural induction. The base cases, p and ⓚ, are trivial; the Boolean cases, $\neg\phi$ and $\phi \wedge \chi$, follow by induction hypothesis; and the modal case, $\Diamond\phi$, is easy to prove. As for the $ⓡ\phi$ case, note that if $\langle W, R, V, M\rangle, w \models ⓡ\phi$, we can delete the edges (s, w) and (w, s) to simulate the storing of w in the memory (if those pairs are not in R' means that $w \in M$) and continue by evaluating the rest of the translation $(\ )'$. □

From the previous theorem, we immediately get:

Theorem 4.2.6. *The satisfiability problem of $\mathcal{ML}(\langle \mathsf{sb}\rangle)$ is undecidable.*

4.2.2 Global Sabotage

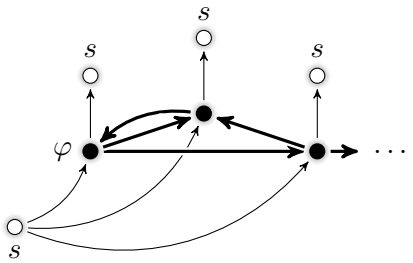
Recall that in Table 2.1 of Chapter 2 we pointed out that the satisfiability problem of the global version of sabotage logic is known to be undecidable. In [LR03a] there is a proof that (multimodal) global sabotage logic is undecidable via a reduction of the Post Correspondence Problem. Here we present an undecidability proof, with a single relation, via a reduction of the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$. We define a computable function $\tau : \mathsf{FORM}_{\mathcal{ML}(ⓡ,ⓚ)} \to \mathsf{FORM}_{\mathcal{ML}(\langle \mathsf{gsb}\rangle)}$ that reduces the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$ to the satisfiability problem of $\mathcal{ML}(\langle \mathsf{gsb}\rangle)$.

As before, the construction of τ proceeds in two steps. First, we define a formula *Struct* that enlarges the original $\mathcal{ML}(ⓡ, ⓚ)$-model with a spy point: for each

successor we enforce a unique edge to an s-state, as we did for the infinite models of Section 3.2.2, that will help us to identify a memorized point. And we also provide a translation Tr from $\mathcal{ML}(ⓡ, ⓚ)$-formulas to $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$-formulas that simulates the ⓡ and ⓚ operators: storing a point in the memory is simulated by deleting the access to the s-successor of the point we want to memorize, and checking whether the current point of evaluation is in the memory is simulated by checking if the arrow of its s-successor has been deleted or not.

Definition 4.2.7. Let φ be an $\mathcal{ML}(ⓡ, ⓚ)$-formula that does not contain the propositional symbol s. We define $\tau(\varphi) = \mathit{Struct} \wedge \mathsf{Tr}(\varphi)$.

A model of $\tau(\varphi)$ for some φ is illustrated below:



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

Let us now see how we can define *Struct* and $\mathsf{Tr}(\varphi)$ to make the reduction.

Definition 4.2.8. Let $\mathcal{M} = \langle W, R, V, M \rangle$ be an $\mathcal{ML}(ⓡ, ⓚ)$-model in which φ is satisfied. We define *Struct* as the conjunction of several properties:

$$
\begin{aligned}
\mathit{Struct} = \quad & s \wedge \Box\neg s && (1') \\
& \wedge \Box(\Diamond s \wedge \langle \mathsf{gsb} \rangle \neg\Diamond s) && (2') \\
& \wedge \Box\Box(\neg s \to \Diamond s \wedge \langle \mathsf{gsb} \rangle \neg\Diamond s) && (3') \\
& \wedge [\mathsf{gsb}](\Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \neg\Diamond s)) \to \Diamond\neg\Diamond s) && (4')
\end{aligned}
$$

Struct defines the structure of our translated models by enlarging the original model with a spy point and particular edges pointing to an s-point. Note that formulas $(1')$ to $(4')$ are analogous to formulas of the infinite models of Section 3.2.2; the main difference is that we don't need to enforce a non-empty set of points ($\Diamond\top$) in $(1')$, and in a similar way we don't need to enforce seriality, irreflexivity, and transitivity. By dropping these properties, the rest of the formulas remain the same: the formula $(2')$ establishes that in one step, there is an s-successor that is unique, and $(3')$ does the same but in two steps. $(4')$ ensures that the evaluation point is linked to every point of the model except to itself and other points where s might hold, i.e., makes the evaluation point a "spy point."

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

Proposition 4.2.9. *Let $\mathcal{M} = \langle W, R, V \rangle$ be a model without a memory and $w \in W$. If $\mathcal{M}, w \models \mathit{Struct}$, then the following properties hold:*

1. *$w \in V(s)$ and for all state $v \in W$ such that $v \neq w$, if $(w, v) \in R$ then there is a **unique** pair $(v, u) \in R$, for some $u \in W$, such that $u \in V(s)$.*
2. *For all states $v \in W$ such that $v \neq w$, we have that if $(w, v) \in R^*$ then $(w, v) \in R$ (w is a spy point).*

Proposition 4.2.9 enumerates the main properties of the spy point: it satisfies the propositional symbol s, and each successor has a unique edge to a point in which s is true – it might be w or another point, it doesn't matter which one it is.

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

Definition 4.2.10. Given φ, we define $\mathsf{Tr}(\varphi) = \Diamond(\varphi)'$, where $(\ )'$ is defined as:

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

$\mathsf{Tr}(\varphi)$ is similar to the translation of the local version of sabotage, the only difference lies in the remember operator. $\textcircled{r}$ is represented by removing the edge from the point we want to memorize to its s-successor: if it has no edge pointing to an s-state ($\neg\Diamond s$), it means that it has already been memorized; otherwise ($\Diamond s$), we use $\langle \mathsf{gsb} \rangle$ to make inaccessible the s-successor of the current point of evaluation and proceed with the translation. $\textcircled{k}$ is represented by checking whether the current point of evaluation has an edge to its s-successor.

Theorem 4.2.11. *Let φ be a formula of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ that does not contain the propositional symbol s. Then, φ is satisfiable if and only if $\tau(\varphi)$ is satisfiable.*

Proof. **(φ is sat $\Leftarrow$ $\tau(\varphi)$ is sat)** Suppose that $\tau(\varphi)$ is satisfiable, i.e., that there exists a model $\mathcal{M} = \langle W, R, V \rangle$ and $s \in W$ such that $\langle W, R, V \rangle, s \models \mathit{Struct}$ and $\langle W, R, V \rangle, s \models \mathsf{Tr}(\varphi)$. We define $\mathcal{M}' = \langle W', R', V', \emptyset \rangle$, where

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

Thus $\mathcal{M}'$ is extracted from the translated model $\mathcal{M}$, and because $\langle W, R, V \rangle, s \models \mathsf{Tr}(\varphi)$, there is $w' \in W'$ such that $(s, w') \in R$ and $\langle W, R, V \rangle, w' \models (\varphi)'$. We will prove:

$$\langle W', R', V', M' \rangle, v' \models \psi \ \text{ iff } \ \langle W, R_{M'}, V \rangle, v' \models (\psi)' \tag{4.4}$$

where $R_{M'} = R \backslash \{(m', u) \mid m' \in M' \wedge u \in V(s)\}$. In particular, when $M' = \emptyset$ we have that $\langle W', R', V', \emptyset \rangle, w' \models \varphi$ iff $\langle W, R, V \rangle, w' \models (\varphi)'$.

We now prove (4.4) by structural induction on ψ. The base cases $\psi = p$ and $\psi = \textcircled{k}$ are analogous to the local version of sabotage, and it is also the same for the inductive cases $\psi = \neg\phi$, $\psi = \phi \wedge \chi$, and $\psi = \Diamond\phi$. Let us prove the interesting case:

$\psi = \textcircled{r}\phi$:

($\Leftarrow$) Suppose that $\langle W, R_{M'}, V \rangle, v' \models (\textcircled{r}\phi)'$. By definition of $(\ )'$ and the semantics we have the conjunction of the following formulas:

$$\begin{array}{ll}
(1) & \langle W, R_{M'}, V \rangle, v' \models \Diamond s \to \langle \mathsf{gsb} \rangle(\neg\Diamond s \wedge (\phi)') \\
(2) & \langle W, R_{M'}, V \rangle, v' \models \neg\Diamond s \to (\phi)'
\end{array}$$

We have to prove that $\langle W', R', V', M' \rangle, v' \models \textcircled{r}\phi$. We will prove it in two parts, first assuming (1) and then assuming (2).

First, assume (1). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \Diamond s$, i.e, that $(v', u) \in R_{M'}$ for some $u \in V(s)$. Then we have that $\langle W, R_{M'}, V\rangle, v' \models \langle \mathsf{gsb}\rangle(\neg\Diamond s \wedge (\phi)')$. By the semantics and our last assumption we have $\langle W, (R_{M'})^{-}_{v'u}, V\rangle, v' \models \neg\Diamond s \wedge (\phi)'$. The first conjunct is trivial because $(v', u) \notin (R_{M'})^{-}_{v'u}$ and by Proposition 4.2.9, so by applying induction hypothesis on the second conjunct, and because we know $v' \in M'$ by definition of $R_{M'}$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$. Hence by the semantics we have $\langle W', R', V', M'\rangle, v' \models \textcircled{r}\phi$.

Finally, assume (2). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s$, i.e, that $(v', u) \notin R_{M'}$ for some $u \in V(s)$, which tell us that v' is in the memory, by definition of $R_{M'}$ and Proposition 4.2.9. Then we have that $\langle W, R_{M'}, V\rangle, v' \models (\phi)'$. By induction hypothesis and because we know $v' \in M'$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$, which is equivalent to $\langle W', R', V', M'\rangle, v' \models \textcircled{r}\phi$ by the semantics.

($\Rightarrow$) Suppose $\langle W', R', V', M'\rangle, v' \models \textcircled{r}\phi$. We have to prove $\langle W, R_{M'}, V\rangle, v' \models (\textcircled{r}\phi)'$, i.e, we have to prove $\langle W, R_{M'}, V\rangle, v' \models \Diamond s \to \langle \mathsf{gsb}\rangle(\neg\Diamond s \wedge (\phi)')$ and $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s \to (\phi)'$. It is easy to prove each part separately (steps are similar to that of the ($\Leftarrow$) direction).

(φ is sat $\Rightarrow$ $\tau(\varphi)$ is sat) Suppose that φ is satisfiable, i.e., that 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 $\mathcal{M}' = \langle W', R', V'\rangle$ as follows:

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

By construction of $\mathcal{M}'$ it is easy to check that $\langle W', R', V'\rangle, s \models \mathit{Struct}$, so it only remains to see that $\langle W', R', V'\rangle, s \models \mathsf{Tr}(\varphi)$. We can verify that

$$\langle W, R, V, M\rangle, w \models \varphi \;\text{ iff }\; \langle W', R'_M, V'\rangle, s \models \mathsf{Tr}(\varphi) \tag{4.5}$$

where $R'_M = R'\backslash\{(m, s) \mid m \in M\}$.

We prove (4.5) by structural induction. The base cases, p and $\textcircled{k}$, are trivial; the Boolean cases, $\neg\phi$ and $\phi \wedge \chi$, follow by induction hypothesis; and the modal case, $\Diamond\phi$, is easy to prove. As for the $\textcircled{r}\phi$ case, note that if $\langle W, R, V, M\rangle, w \models \textcircled{r}\phi$, we can delete the edge (w, s) to simulate the storing of w in the memory (if this pair is not in R' means that $w \in M$) and continue by evaluating the rest of the translation $(\ )'$. □

From the previous theorem, we immediately get:

Theorem 4.2.12. *The satisfiability problem of $\mathcal{ML}(\langle \mathsf{gsb}\rangle)$ is undecidable.*

4.3 Bridge Logic

4.3.1 Local Bridge

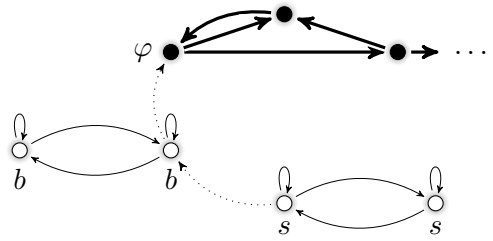
We define a computable function $\tau : \mathsf{FORM}_{\mathcal{ML}(\textcircled{r},\textcircled{k})} \to \mathsf{FORM}_{\mathcal{ML}(\langle \mathsf{br}\rangle)}$ that reduces the satisfiability problem of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ to the satisfiability problem of $\mathcal{ML}(\langle \mathsf{br}\rangle)$.

As usual, the construction of τ proceeds in two steps. First, we enforce some constraints on the structure of our translated models with a formula *Struct* that enlarges the original $\mathcal{ML}(\textcircled{r}, \textcircled{k})$-model with two sets of points: a set of points in

which the propositional symbol s is true, and a set of points in which the propositional symbol b is true. As for the s-set, it is analogous to the one of the infinite models of Section 3.3.1, i.e., the points are all connected among them (one of them being the spy point) and disconnected with the points of the $\mathcal{ML}(ⓡ, ⓚ)$-model. As for the b-set, it has the same properties as the s-set except that b is true instead of s. In this second set one point is used as a "bridge" to connect the spy point with the point of the $\mathcal{ML}(ⓡ, ⓚ)$-model in which the formula we want to translate is satisfied (the need for the b-set will become clear later when we present the definition of $\tau(\varphi)$). And we also provide a translation Tr from $\mathcal{ML}(ⓡ, ⓚ)$-formulas to $\mathcal{ML}(\langle \mathsf{br} \rangle)$-formulas that simulates the ⓡ and ⓚ operators: storing a point in the memory is simulated by adding edges back and forth from the state we want to memorize to the spy point, and checking whether the current point of evaluation is in the memory is simulated by checking if an edge to the spy point exists.

Definition 4.3.1. Let φ be an $\mathcal{ML}(ⓡ, ⓚ)$-formula that does not contain the propositional symbols s and b. We define $\tau(\varphi) = \mathit{Struct} \wedge \mathsf{Tr}(\varphi)$.

A model of $\tau(\varphi)$ for some φ is illustrated below:



In this picture, the black points and thick lines represent the model of the initial memory logic formula that can be extracted from the whole model; the dotted lines represent the edges created with $\langle \mathsf{br} \rangle$.

Let us now see how we can define *Struct* and $\mathsf{Tr}(\varphi)$ to make the reduction.

Definition 4.3.2. Let $\mathcal{M} = \langle W, R, V, M \rangle$ be an $\mathcal{ML}(ⓡ, ⓚ)$-model in which φ is satisfied. We define *Struct* as the conjunction of several properties:

$$
\begin{array}{lll}
\mathit{Struct} = & s \wedge [\mathsf{br}]\neg s & (1') \\
& \wedge \Box[\mathsf{br}]\neg s & (2') \\
& \wedge \Box\Box s & (3') \\
& \wedge \Box[\mathsf{br}]\Box\neg s & (4') \\
& \wedge \Box\neg b & (5') \\
& \wedge \langle \mathsf{br} \rangle (b \wedge [\mathsf{br}]\neg b & (6') \\
& \qquad \wedge \Box[\mathsf{br}]\neg b & \\
& \qquad \wedge \Box\Box b & \\
& \qquad \wedge \Box[\mathsf{br}]\Box\neg b & \\
& \qquad \wedge \Box\neg s) &
\end{array}
$$

Struct defines the structure of our translated models by enlarging the original model with two sets of points: a set of "spy points" in which s is true, and a set of "bridge points" in which b is true. Note that formulas $(1')$ to $(4')$ are analogous to formulas of the infinite models of Section 3.3.1; the main difference is that we don't need to enforce a non-empty set of points ($\langle \mathsf{br} \rangle \top$) in $(1')$, and in a similar way we don't need to enforce seriality, irreflexivity, and transitivity. By dropping these properties, formulas $(1')$-$(4')$ remain the same: the formula $(1')$ establishes that

the evaluation point cannot access the points of the $\mathcal{ML}(ⓡ, ⓚ)$-model (and that it satisfies s and that it is reflexive); $(2')$ ensures that there is no unconnected s-state; $(3')$ ensures that all successors have only s-successors; and $(4')$ ensures that there no edges from $\neg s$-states into the connected component satisfying s. We added $(5')$ to ensure that no successor of an element of the s-set satisfies the new propositional symbol b. And we also added $(6')$ which establishes that there is a totally connected component of states satisfying b that is unreachable from the s-component. This set of b-states is described exactly as for the set of s-states: observe that in $(6')$, what is in the scope of the $\langle \mathsf{br} \rangle$ operator is the conjunction of formulas $(1')$ to $(5')$, but replacing b for s and vice versa.

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

Proposition 4.3.3. *Let $\mathcal{M} = \langle W, R, V \rangle$ be a model without a memory and $w \in W$. If $\mathcal{M}, w \models Struct$, 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 $(v, w) \notin R$.*

Proposition 4.3.3 enumerates the main properties of the spy point and the two sets we define: the spy point satisfies s but does not satisfies b, and both the s-set and the b-set are totally connected, with the members of one set being disconnected from the members of the other set.

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

Definition 4.3.4. Given φ, we define $\mathsf{Tr}(\varphi) = \langle \mathsf{br} \rangle (b \wedge \langle \mathsf{br} \rangle (\neg s \wedge (\varphi)'))$, where $(\ )'$ is defined as:

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

$\mathsf{Tr}(\varphi)$ places the translation $(\ )'$ of the memory logic formula φ in an inaccessible $\mathcal{ML}(ⓡ, ⓚ)$-state. Note that it does not create an edge from the spy point to the point of the $\mathcal{ML}(ⓡ, ⓚ)$-model directly. Instead, it creates an edge to an inaccessible b-state (which acts as a "bridge" between the spy point and the $\mathcal{ML}(ⓡ, ⓚ)$-model), and from there, creates another edge to the point of the $\mathcal{ML}(ⓡ, ⓚ)$-model that satisfies φ. Then, we evaluate the translation $(\ )'$ on φ. Boolean and modal cases are obvious. ⓡ is represented by creating arrows from the state we want to memorize to the spy point, and from the spy point to the memorized state. Note how the translation behaves: if the point has already been memorized ($\Diamond s$), then nothing needs to be done and the translation continues; otherwise ($\neg\Diamond s$), we make one point of the s-component accessible using $\langle \mathsf{br} \rangle$, and this particular s-point must be the one that has a b-successor. (Thus, even tough we might have several points satisfying s in

the connected component, we use only one of them as *the* spy point.) And then, from the spy point, we create an arrow to the $\mathcal{ML}(ⓡ, ⓚ)$-state we want to memorize. ⓚ is represented by checking whether there is an edge pointing to the spy point: if such an edge does exist it means that the current point of evaluation is in the memory.

Theorem 4.3.5. *Let φ be a formula of $\mathcal{ML}(ⓡ, ⓚ)$ that does not contain the propositional symbols s and b. Then, φ is satisfiable if and only if $\tau(\varphi)$ is satisfiable.*

Proof. **(φ is sat $\Leftarrow$ $\tau(\varphi)$ is sat)** Suppose that $\tau(\varphi)$ is satisfiable, i.e., that there exists a model $\mathcal{M} = \langle W, R, V\rangle$ and $s \in W$ such that $\langle W, R, V\rangle, s \models \mathit{Struct}$ and $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$. We define $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$, where

$$\begin{array}{ll} W' & = W \backslash (V(s) \cup V(b)) \\ R' & = R \cap (W' \times W') \\ V'(p) & = V(p) \cap W' \quad \text{for } p \in \mathsf{PROP} \end{array}$$

Thus $\mathcal{M}'$ is extracted from the translated model $\mathcal{M}$, and because $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$, there are $b, w' \in W$ such that $b \in V(b)$, $w' \notin V(s)$, $\{(s, b), (b, w')\} \subseteq R$, and $\langle W, R, V\rangle, w' \models (\varphi)'$. We will prove:

$$\langle W', R', V', M'\rangle, v' \models \psi \;\text{ iff }\; \langle W, R_{M'}, V\rangle, v' \models (\psi)' \tag{4.6}$$

where $R_{M'} = R \cup \{(m', s), (s, m') \mid m' \in M'\}$. In particular, when $M' = \emptyset$ we have that $\langle W', R', V', \emptyset\rangle, w' \models \varphi$ iff $\langle W, R, V\rangle, w' \models (\varphi)'$.

We now prove (4.6) by structural induction on ψ. Boolean and modal cases are easy, and it is also the case for ⓚ. Let us prove the interesting case:

$\psi = ⓡ\phi$:

($\Leftarrow$) Suppose that $\langle W, R_{M'}, V\rangle, v' \models (ⓡ\phi)'$. By definition of $(\ )'$ and the semantics we have the conjunction of the following formulas:

(1) $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s \to \langle \mathsf{br}\rangle(s \wedge \Diamond b \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)'))$
(2) $\langle W, R_{M'}, V\rangle, v' \models \Diamond s \to (\phi)'$

We have to prove that $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$. We will prove it in two parts, first assuming (1) and then assuming (2).

First, assume (1). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s$, i.e, that $(v', u) \notin R_{M'}$ for some $u \in V(s)$. Then we have that $\langle W, R_{M'}, V\rangle, v' \models \langle \mathsf{br}\rangle(s \wedge \Diamond b \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)'))$. By the semantics and our last assumption we have $\langle W, (R_{M'})^+_{v's}, V\rangle, s \models s \wedge \Diamond b \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)')$. It is easy to see that $\langle W, (R_{M'})^+_{v's}, V\rangle, s \models s$, so let us check $\langle W, (R_{M'})^+_{v's}, V\rangle, s \models \Diamond b \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)')$. Then $\langle W, (R_{M'})^+_{v's}, V\rangle, s \models \Diamond b$ because by definition of Tr the evaluation started at s by adding and edge to a b-state (that does not satisfy s by Proposition 4.3.3). It remains to see that $\langle W, (R_{M'})^+_{v's}, V\rangle, s \models \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)')$. By the semantics and because $(s, v') \notin (R_{M'})^+_{v's}$ by Proposition 4.3.3, we have $\langle W, (R_{M'})^+_{v's,sv'}, V\rangle, v' \models \Diamond s \wedge (\phi)'$. The first conjunct is trivial by Proposition 4.3.3 and because $(v', s) \in (R_{M'})^+_{v's,sv'}$, so by applying induction hypothesis on the second conjunct, and because we know $v' \in M'$ by definition of $R_{M'}$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$. Hence by the semantics we have $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$.

Finally, assume (2). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \Diamond s$, i.e, that $(v', s) \in R_{M'}$, which in turn also tell us that $(s, v') \in R_{M'}$, by definition of $R_{M'}$ and Proposition 4.3.3. Then we have that $\langle W, R_{M'}, V\rangle, v' \models (\phi)'$. By induction

hypothesis and because we know $v' \in M'$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$, which is equivalent to $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$ by the semantics.

($\Rightarrow$) Suppose $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$. We have to prove $\langle W, R_{M'}, V\rangle, v' \models (ⓡ\phi)'$, i.e, we have to prove $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s \to \langle \mathsf{br}\rangle(s \wedge \Diamond b \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)'))$ and $\langle W, R_{M'}, V\rangle, v' \models \Diamond s \to (\phi)'$. It is easy to prove each part separately (steps are similar to that of the ($\Leftarrow$) direction).

(φ is sat $\Rightarrow$ $\tau(\varphi)$ is sat) Suppose that φ is satisfiable, i.e., that 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 and b be states that do not belong to W. Then we can define $\mathcal{M}' = \langle W', R', V'\rangle$ as follows:

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

By construction of $\mathcal{M}'$ it is easy to check that $\langle W', R', V'\rangle, s \models \mathit{Struct}$, so it only remains to see that $\langle W', R', V'\rangle, s \models \mathsf{Tr}(\varphi)$. We can verify that

$$\langle W, R, V, M\rangle, w \models \varphi \;\text{ iff }\; \langle W', R'_M, V'\rangle, s \models \mathsf{Tr}(\varphi) \tag{4.7}$$

where $R'_M = R' \cup \{(m, s), (s, m) \mid m \in M\}$.

We prove (4.7) by structural induction. The base cases, p and ⓚ, are trivial; the Boolean cases, $\neg\phi$ and $\phi \wedge \chi$, follow by induction hypothesis; and the modal case, $\Diamond\phi$, is easy to prove. As for the $ⓡ\phi$ case, note that if $\langle W, R, V, M\rangle, w \models ⓡ\phi$, we can create the edges (w, s) and (s, w) to simulate the storing of w in the memory (if those pairs are already in R' means that $w \in M$) and continue by evaluating the rest of the translation $(\ )'$. □

From the previous theorem, we immediately get:

Theorem 4.3.6. *The satisfiability problem of $\mathcal{ML}(\langle \mathsf{br}\rangle)$ is undecidable.*

4.3.2 Global Bridge

We define a computable function $\tau : \mathsf{FORM}_{\mathcal{ML}(ⓡ,ⓚ)} \to \mathsf{FORM}_{\mathcal{ML}(\langle \mathsf{gbr}\rangle)}$ that reduces the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$ to the satisfiability problem of $\mathcal{ML}(\langle \mathsf{gbr}\rangle)$.

As usual, the construction of τ proceeds in two steps. First, we define a formula *Struct* that enlarges the original $\mathcal{ML}(ⓡ, ⓚ)$-model with a spy point: for each successor we enforce the non-existence of s-successors as we did for the infinite models of Section 3.3.2. And we also provide a translation Tr from $\mathcal{ML}(ⓡ, ⓚ)$-formulas to $\mathcal{ML}(\langle \mathsf{gbr}\rangle)$-formulas that simulates the ⓡ and ⓚ operators: storing a point in the memory is simulated by creating an arrow to an s-state from the point we want to memorize, and checking whether the current point of evaluation is in the memory is simulated by checking if an arrow to an s-state has been created or not.

Definition 4.3.7. Let φ be an $\mathcal{ML}(ⓡ, ⓚ)$-formula that does not contain the propositional symbol s. We define $\tau(\varphi) = \mathit{Struct} \wedge \mathsf{Tr}(\varphi)$.

A model of $\tau(\varphi)$ for some φ is illustrated in Figure 4.4. The black points and thick lines represent the model of the initial memory logic formula that can be extracted from the whole model.

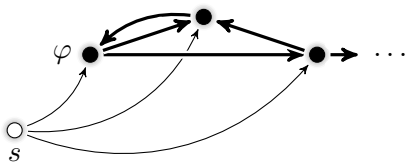


Figure 4.4: A model of $\tau(\varphi)$ for some φ.

Let us now see how we can define *Struct* and $\mathsf{Tr}(\varphi)$ to make the reduction.

Definition 4.3.8. Let $\mathcal{M} = \langle W, R, V, M\rangle$ be an $\mathcal{ML}(ⓡ, ⓚ)$-model in which φ is satisfied. We define *Struct* as the conjunction of several properties:

$$\begin{array}{lll} \mathit{Struct} = & s \wedge \Box\neg s & (1') \\ & \wedge\ \Box\Box\neg s & (2') \\ & \wedge\ \Box\Box(\neg s \to \Box\neg s) & (3') \\ & \wedge\ [\mathsf{gbr}](\Diamond(\neg\Diamond s \wedge \Diamond\Diamond s) \ \to\ \Diamond\Diamond s) & (4') \end{array}$$

Struct defines the structure of our translated models by enlarging the original model with a spy point. Note that formulas $(1')$ to $(4')$ are analogous to formulas of the infinite models of Section 3.3.2; the main difference is that we don't need to enforce a non-empty set of points ($\Diamond\top$) in $(1')$, and in a similar way we don't need to enforce seriality, irreflexivity, and transitivity. By dropping these properties, the rest of the formulas remain the same: the formula $(2')$ establishes that in one step, there are no s-successors, and $(3')$ does the same but in two steps. $(4')$ ensures that the evaluation point is linked to every point of the model except to itself, i.e., makes the evaluation point a "spy point."

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

Proposition 4.3.9. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model without a memory and $w \in W$. If $\mathcal{M}, w \models \mathit{Struct}$, then the following properties hold:*

1. *$w \in V(s)$ and for all state $v \in W$ such that $v \neq w$, if $(w, v) \in R$ then **there is not any** pair of the form $(v, u) \in R$, for some $u \in W$, such that $u \in V(s)$.*
2. *For all states $v \in W$ such that $v \neq w$, we have that if $(w, v) \in R^*$ then $(w, v) \in R$ (w is a spy point).*

Proposition 4.3.9 enumerates the main properties of the spy point: it satisfies the propositional symbol s, and for each successor there are no edges to s-points.

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

Definition 4.3.10. Given φ, we define $\mathsf{Tr}(\varphi) = \Diamond(\varphi)'$, where $(\ )'$ is defined as:

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

$\mathsf{Tr}(\varphi)$ is similar to the translation of the global version of sabotage: we create an edge to an s-state instead of delete it. ⓡ is represented by creating an edge to

an s-point from the state we want to memorize: if it has an s-successor ($\Diamond s$), then nothing needs to be done and the translation continues; otherwise ($\neg\Diamond s$), we add an edge pointing to a state that satisfies s from the current point of evaluation and proceed with the translation. Ⓚ is represented by checking whether the current point of evaluation has an edge to an s-point.

Theorem 4.3.11. *Let φ be a formula of $\mathcal{ML}(Ⓡ, Ⓚ)$ that does not contain the propositional symbol s. Then, φ is satisfiable if and only if $\tau(\varphi)$ is satisfiable.*

Proof. **(φ is sat $\Leftarrow$ $\tau(\varphi)$ is sat)** Suppose that $\tau(\varphi)$ is satisfiable, i.e., that there exists a model $\mathcal{M} = \langle W, R, V\rangle$ and $s \in W$ such that $\langle W, R, V\rangle, s \models \mathit{Struct}$ and $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$. We define $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$, where

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

Thus $\mathcal{M}'$ is extracted from the translated model $\mathcal{M}$, and because $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$, there is $w' \in W'$ such that $(s, w') \in R$ and $\langle W, R, V\rangle, w' \models (\varphi)'$. We will prove:

$$\langle W', R', V', M'\rangle, v' \models \psi \;\text{ iff }\; \langle W, R_{M'}, V\rangle, v' \models (\psi)' \tag{4.8}$$

where $R_{M'} = R \cup \{(m', u) \mid m' \in M' \wedge u \in V(s)\}$. In particular, when $M' = \emptyset$ we have that $\langle W', R', V', \emptyset\rangle, w' \models \varphi$ iff $\langle W, R, V\rangle, w' \models (\varphi)'$.

We now prove (4.8) by structural induction on ψ. Boolean and modal cases are easy, and it is also the case for Ⓚ. Let us prove the interesting case:

$\boldsymbol{\psi = Ⓡ\phi}$:

($\Leftarrow$) Suppose that $\langle W, R_{M'}, V\rangle, v' \models (Ⓡ\phi)'$. By definition of $(\ )'$ and the semantics we have the conjunction of the following formulas:

$$\begin{array}{ll} (1) & \langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s \to \langle \mathsf{gbr}\rangle(\Diamond s \wedge (\phi)') \\ (2) & \langle W, R_{M'}, V\rangle, v' \models \Diamond s \to (\phi)' \end{array}$$

We have to prove that $\langle W', R', V', M'\rangle, v' \models Ⓡ\phi$. We will prove it in two parts, first assuming (1) and then assuming (2).

First, assume (1). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s$, i.e, that $(v', u) \notin R_{M'}$ for some $u \in V(s)$. Then we have that $\langle W, R_{M'}, V\rangle, v' \models \langle \mathsf{gbr}\rangle(\Diamond s \wedge (\phi)')$. By the semantics and our last assumption we have $\langle W, (R_{M'})^{+}_{v'u}, V\rangle, v' \models \Diamond s \wedge (\phi)'$. The first conjunct is trivial because $(v', u) \in (R_{M'})^{+}_{v'u}$ and by Proposition 4.3.9, so by applying induction hypothesis on the second conjunct, and because we know $v' \in M'$ by definition of $R_{M'}$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$. Hence by the semantics we have $\langle W', R', V', M'\rangle, v' \models Ⓡ\phi$.

Finally, assume (2). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \Diamond s$, i.e, that $(v', u) \in R_{M'}$ for some $u \in V(s)$, which tell us that v' is in the memory, by definition of $R_{M'}$ and Proposition 4.3.9. Then we have that $\langle W, R_{M'}, V\rangle, v' \models (\phi)'$. By induction hypothesis and because we know $v' \in M'$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$, which is equivalent to $\langle W', R', V', M'\rangle, v' \models Ⓡ\phi$ by the semantics.

($\Rightarrow$) Suppose $\langle W', R', V', M'\rangle, v' \models Ⓡ\phi$. We have to prove $\langle W, R_{M'}, V\rangle, v' \models (Ⓡ\phi)'$, i.e, we have to prove $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond s \to \langle \mathsf{gbr}\rangle(\Diamond s \wedge (\phi)')$ and $\langle W, R_{M'}, V\rangle, v' \models \Diamond s \to (\phi)'$. It is easy to prove each part separately (steps are similar to that of the ($\Leftarrow$) direction).

(φ is sat $\Rightarrow$ $\tau(\varphi)$ is sat) Suppose that φ is satisfiable, i.e., that 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 $\mathcal{M}' = \langle W', R', V'\rangle$ as follows:

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

By construction of $\mathcal{M}'$ it is easy to check that $\langle W', R', V'\rangle, s \models \mathit{Struct}$, so it only remains to see that $\langle W', R', V'\rangle, s \models \mathsf{Tr}(\varphi)$. We can verify that

$$\langle W, R, V, M\rangle, w \models \varphi \;\text{ iff }\; \langle W', R'_M, V'\rangle, s \models \mathsf{Tr}(\varphi) \tag{4.9}$$

where $R'_M = R' \cup \{(m, s) \mid m \in M\}$.

We prove (4.9) by structural induction. The base cases, p and $\textcircled{k}$, are trivial; the Boolean cases, $\neg\phi$ and $\phi \wedge \chi$, follow by induction hypothesis; and the modal case, $\Diamond\phi$, is easy to prove. As for the $\textcircled{r}\phi$ case, note that if $\langle W, R, V, M\rangle, w \models \textcircled{r}\phi$, we can create the edge (w, s) to simulate the storing of w in the memory (if this pair is already in R' means that $w \in M$) and continue by evaluating the rest of the translation $(\ )'$. □

From the previous theorem, we immediately get:

Theorem 4.3.12. *The satisfiability problem of $\mathcal{ML}(\langle \mathsf{gbr}\rangle)$ is undecidable.*

4.4 Swap Logic

4.4.1 Local Swap

Recall that in Table 2.1 of Chapter 2 we pointed out that the satisfiability problem of the local version of swap logic is known to be undecidable. In [AFH14] there is a proof of undecidability for local swap logic via a reduction of the satisfiability problem of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$. The proof we present here is an adaptation of the proof that appears in that paper. We define a computable function $\tau : \mathsf{FORM}_{\mathcal{ML}(\textcircled{r},\textcircled{k})} \to \mathsf{FORM}_{\mathcal{ML}(\langle \mathsf{sw}\rangle)}$ that reduces the satisfiability problem of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ to the satisfiability problem of $\mathcal{ML}(\langle \mathsf{sw}\rangle)$.

As usual, the construction of τ proceeds in two steps. First, we define a formula *Struct* that enlarges the original $\mathcal{ML}(\textcircled{r}, \textcircled{k})$-model with a spy point and with "switches": we enforce a spy point as we did for the infinite models of Section 3.4, and additionally, for each point of the $\mathcal{ML}(\textcircled{r}, \textcircled{k})$-model we enforce "switches," i.e., special edges whose position ("off" by default, and "on" if the direction of the edge has been swap around) will help us to identify a memorized point. And we also provide a translation Tr from $\mathcal{ML}(\textcircled{r}, \textcircled{k})$-formulas to $\mathcal{ML}(\langle \mathsf{sw}\rangle)$-formulas that simulates the $\textcircled{r}$ and $\textcircled{k}$ operators: storing a point in the memory is simulated by swapping the switch of the point we want to memorize, and checking whether the current point of evaluation is in the memory is simulated by checking the position of the switch.

Definition 4.3.1. Let φ be an $\mathcal{ML}(\textcircled{r}, \textcircled{k})$-formula that does not contain the propositional symbols s and sw. We define $\tau(\varphi) = \mathit{Struct} \wedge \mathsf{Tr}(\varphi)$.

A model of $\tau(\varphi)$ for some φ is illustrated in Figure 4.5. The black points and thick lines represent the model of the initial memory logic formula that can be extracted from the whole model.

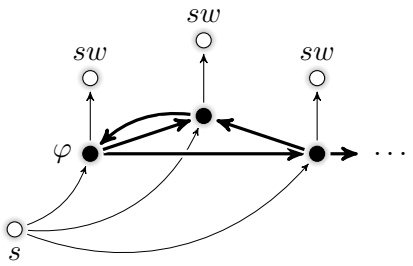


Figure 4.5: A model of $\tau(\varphi)$ for some φ.

Let us now see how we can define *Struct* and $\mathsf{Tr}(\varphi)$ to make the reduction.

Definition 4.4.2. Let $\mathcal{M} = \langle W, R, V, M\rangle$ be an $\mathcal{ML}(ⓡ, ⓚ)$-model in which φ is satisfied. We define *Struct* as the conjunction of several properties:

$$
\begin{aligned}
\mathit{Struct} = \quad & s \wedge \Box^{(6)}\neg s && (1')\\
& \wedge\ \neg sw \wedge \Box\neg sw \wedge \Box(\Diamond(sw \wedge \Box\bot) \wedge [\mathsf{sw}](sw \to \Box\neg\Diamond sw)) && (2')\\
& \wedge\ [\mathsf{sw}][\mathsf{sw}]\Box\Box\Box\Box(sw \to \Box\bot) && (3')\\
& \wedge\ [\mathsf{sw}][\mathsf{sw}](\neg s \wedge \neg sw \to \langle\mathsf{sw}\rangle(sw \wedge \Diamond\Diamond\Diamond(s \wedge \Diamond\neg\Diamond sw)) && (4')
\end{aligned}
$$

Struct defines the structure of our translated models by enlarging the original model with a spy point and "switches." The propositional symbol sw represents "switch points" that 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 6 steps from there. $(2')$ initialises 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 to itself and the switch points, i.e., makes the evaluation point a "spy point."

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

Proposition 4.4.3. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model without a memory and $w \in W$. If $\mathcal{M}, w \models$ Struct, then the following properties hold:*

1. *$w \in V(s)$ and for all state $v \in W$ such that $v \neq w$, if $(w, v) \in R^*$ then $v \notin V(s)$ within 6 steps.*

2. *For all state $v \in W$ such that $v \neq w$, if $(w, v) \in R$ then there is a **unique** edge $(v, u) \in R$, for some $u \in W$, such that $u \in V(sw)$.*

3. *For all states $v \in W$ such that $v \neq w$, if $(w, v) \in R^*$ then $(w, v) \in R$ (w is a spy point).*

Proposition 4.4.3 enumerates the main properties of the spy point: it is the only point that satisfies the propositional symbol s within 6 steps, and each successor has a unique edge to a point in which the propositional symbol sw is true.

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

Definition 4.4.4. Given φ, we define $\mathsf{Tr}(\varphi) = \Diamond(\varphi)'$, where $(\ )'$ is defined as:

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

$\mathsf{Tr}(\varphi)$ places the translation $(\ )'$ of the memory logic formula φ right after the evaluation point. Then, we evaluate the translation $(\ )'$ on φ. Boolean cases are obvious. For the diamond case, $\Diamond\psi$ is satisfied if there is a successor v where ψ holds, but we must ensure that v is a point of the original $\mathcal{ML}(\textcircled{r}, \textcircled{k})$-model, so v cannot satisfy s nor sw. $\textcircled{r}$ is represented by swapping the edge between the point we want to memorize and its switch point. It is important that switch points do not have successors and that they have exactly one predecessor. This ensures that the path taken by $(\textcircled{r}\psi)'$ correctly comes back to the same point of the model. $\textcircled{k}$ is represented by checking whether the current point has an edge to its switch point: if such edge does not exist it means that it has already been swapped and that the current point of evaluation is in the memory.

Theorem 4.4.5. *Let φ be a formula of $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ that does not contain the propositional symbols s and sw. Then, φ is satisfiable if and only if $\tau(\varphi)$ is satisfiable.*

Proof. **(φ is sat $\Leftarrow$ $\tau(\varphi)$ is sat)** Suppose that $\tau(\varphi)$ is satisfiable, i.e., that there exists a model $\mathcal{M} = \langle W, R, V\rangle$ and $s \in W$ such that $\langle W, R, V\rangle, s \models \mathit{Struct}$ and $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$. We define $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$, where

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

Thus $\mathcal{M}'$ is extracted from the translated model $\mathcal{M}$, and because $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$, there is $w' \in W'$ such that $(s, w') \in R$ and $\langle W, R, V\rangle, w' \models (\varphi)'$. We will prove:

$$\langle W', R', V', M'\rangle, v' \models \psi \ \text{ iff } \ \langle W, R_{M'}, V\rangle, v' \models (\psi)' \tag{4.10}$$

where $R_{M'} = (R \backslash S^{-1}) \cup S$ with $S = \{(v, m') \mid m' \in M' \wedge v \in V(sw)\}$. In particular, when $M' = \emptyset$ we have that $\langle W', R', V', \emptyset\rangle, w' \models \varphi$ iff $\langle W, R, V\rangle, w' \models (\varphi)'$.

We now prove (4.10) by structural induction on ψ. Boolean and modal cases are easy, and it is also the case for $\textcircled{k}$. Let us prove the interesting case:

$\boldsymbol{\psi = \textcircled{r}\phi}$:

($\Leftarrow$) Suppose that $\langle W, R_{M'}, V\rangle, v' \models (\textcircled{r}\phi)'$. By definition of $(\ )'$ and the semantics we have the conjunction of the following formulas:

$$\begin{array}{ll}
(1) & \langle W, R_{M'}, V\rangle, v' \models \Diamond sw \to \langle \mathsf{sw} \rangle (sw \wedge \Diamond(\phi)') \\
(2) & \langle W, R_{M'}, V\rangle, v' \models \neg\Diamond sw \to (\phi)'
\end{array}$$

We have to prove that $\langle W', R', V', M'\rangle, v' \models \textcircled{r}\phi$. We will prove it in two parts, first assuming (1) and then assuming (2).

First, assume (1). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \Diamond sw$, i.e, that $(v', u) \in R_{M'}$ for some $u \in V(sw)$. Then we have that $\langle W, R_{M'}, V\rangle, v' \models \langle \mathsf{sw} \rangle (sw \wedge \Diamond(\phi)')$. By

the semantics and our last assumption we have $\langle W, (R_{M'})^*_{uv'}, V\rangle, u \models sw \wedge \Diamond(\phi)'$. The first conjunct is trivial because $u \in V(sw)$ and by Proposition 4.4.3, so by applying induction hypothesis on the second conjunct, and because we know $v' \in M'$ by definition of $R_{M'}$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$. Hence by the semantics we have $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$.

Finally, assume (2). Now suppose $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond sw$, i.e, that $(v', u) \notin R_{M'}$ for some $u \in V(sw)$, which in turn also tell us that $(u, v') \in R_{M'}$, by definition of $R_{M'}$ and Proposition 4.4.3, and therefore v' is in the memory. Then we have that $\langle W, R_{M'}, V\rangle, v' \models (\phi)'$. By induction hypothesis and because we know $v' \in M'$, we have $\langle W', R', V', M' \cup \{v'\}\rangle, v' \models \phi$, which is equivalent to $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$ by the semantics.

($\Rightarrow$) Suppose $\langle W', R', V', M'\rangle, v' \models ⓡ\phi$. We have to prove $\langle W, R_{M'}, V\rangle, v' \models (ⓡ\phi)'$, i.e, we have to prove $\langle W, R_{M'}, V\rangle, v' \models \Diamond sw \to \langle \mathsf{sw}\rangle(sw \wedge \Diamond(\phi)')$ and $\langle W, R_{M'}, V\rangle, v' \models \neg\Diamond sw \to (\phi)'$. It is easy to prove each part separately (steps are similar to that of the ($\Leftarrow$) direction).

(φ is sat $\Rightarrow$ $\tau(\varphi)$ is sat) Suppose that φ is satisfiable, i.e., that 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 s 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:

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

By construction of $\mathcal{M}'$ it is easy to check that $\langle W', R', V'\rangle, s \models \mathit{Struct}$, so it only remains to see that $\langle W', R', V'\rangle, s \models \mathsf{Tr}(\varphi)$. We can verify that

$$\langle W, R, V, M\rangle, w \models \varphi \;\text{ iff }\; \langle W', R'_M, V'\rangle, s \models \mathsf{Tr}(\varphi) \tag{4.11}$$

where $R'_M = (R' \backslash S^{-1}) \cup S$ with $S = \{(sw(m), m) \mid m \in M\}$.

We prove (4.11) by structural induction. The base cases, p and ⓚ, are trivial; the Boolean cases, $\neg\phi$ and $\phi \wedge \chi$, follow by induction hypothesis; and the modal case, $\Diamond\phi$, is easy to prove. As for the $ⓡ\phi$ case, note that if $\langle W, R, V, M\rangle, w \models ⓡ\phi$, we can swap the edge $(w, sw(w))$ to simulate the storing of w in the memory (if $(w, sw(w)) \notin R'$ and $(sw(w), w) \in R'$ means that $w \in M$) and continue by evaluating the rest of the translation $(\ )'$. □

From the previous theorem, we immediately get:

Theorem 4.4.6. *The satisfiability problem of $\mathcal{ML}(\langle \mathsf{sw}\rangle)$ is undecidable.*

4.4.2 Global Swap

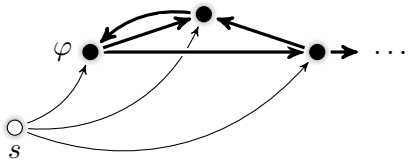
We define a computable function $\tau : \mathsf{FORM}_{\mathcal{ML}(ⓡ,ⓚ)} \to \mathsf{FORM}_{\mathcal{ML}(\langle \mathsf{gsw}\rangle)}$ that reduces the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$ to the satisfiability problem of $\mathcal{ML}(\langle \mathsf{gsw}\rangle)$.

As usual, the construction of τ proceeds in two steps. First, we enforce some constraints on the structure of our translated models with a formula *Struct* that enlarges the original $\mathcal{ML}(ⓡ, ⓚ)$-model with a spy point as we did for the infinite models of Section 3.4. And we also provide a translation Tr from $\mathcal{ML}(ⓡ, ⓚ)$-formulas

to $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$-formulas that simulates the ⓡ and ⓚ operators: storing a point in the memory is simulated by swapping the edge that connects the spy point with point we want to memorize, and checking whether the current point of evaluation is in the memory is simulated by checking if there is an edge pointing to the spy point.

Definition 4.4.7. Let φ be an $\mathcal{ML}(ⓡ, ⓚ)$-formula that does not contain the propositional symbol s. We define $\tau(\varphi) = \mathit{Struct} \wedge \mathsf{Tr}(\varphi)$.

A model of $\tau(\varphi)$ for some φ is illustrated below:



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

Let us now see how we can define *Struct* and $\mathsf{Tr}(\varphi)$ to make the reduction.

Definition 4.4.8. Let $\mathcal{M} = \langle W, R, V, M \rangle$ be an $\mathcal{ML}(ⓡ, ⓚ)$-model in which φ is satisfied. We define *Struct* as the conjunction of several properties:

$$\begin{array}{rll} \mathit{Struct} = & s \wedge \Box^{(7)} \neg s & (1') \\ & \wedge \Box\Box[\mathsf{gsw}][\mathsf{gsw}]\Box\Box(s \to \Diamond\Diamond\Diamond s) & (2') \end{array}$$

Struct defines the structure of our translated models by enlarging the original model with a spy point. Note that formulas $(1')$, $(2')$ are analogous to formulas of the infinite models of Section 3.4; the main difference is that we don't need to enforce a non-empty set of points ($\Diamond\top$) and seriality ($\Box\Diamond\top$) in $(1')$, and in a similar way we don't need to enforce irreflexivity and transitivity. By removing these restrictions, the formulas remain the same: $(2')$ ensures that the evaluation point is linked to every point of the model except to itself, i.e., makes the evaluation point a "spy point."

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

Proposition 4.4.9. *Let $\mathcal{M} = \langle W, R, V \rangle$ be a model without a memory and $w \in W$. If $\mathcal{M}, w \models \mathit{Struct}$, then the following properties hold:*

1. *$w \in V(s)$ and for all state $v \in W$ such that $v \neq w$, if $(w, v) \in R^*$ then $v \notin V(s)$ within 7 steps.*
2. *For all states $v \in W$ such that $v \neq w$, if $(w, v) \in R^*$ then $(w, v) \in R$ (w is a spy point).*

Proposition 4.4.9 enumerates the main properties of the spy point: it is the only point that satisfies the propositional symbol s within 7 steps.

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

Definition 4.4.10. Given φ, we define $\mathsf{Tr}(\varphi) = \Diamond(\varphi)'$, where $(\ )'$ is defined as:

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

$\mathsf{Tr}(\varphi)$ is similar to the translation of the global version of bridge: just replace $\langle \mathsf{gsw} \rangle$ for $\langle \mathsf{gbr} \rangle$ and it all works. ⓡ is represented by swapping the edge that connects the spy point with the point we want to memorize (instead of creating an arrow to an s-point), and ⓚ is represented by checking if the current point of evaluation can access the spy point.

Theorem 4.4.11. *Let φ be a formula of $\mathcal{ML}(ⓡ, ⓚ)$ that does not contain the propositional symbol s. Then, φ is satisfiable if and only if $\tau(\varphi)$ is satisfiable.*

Proof. **(φ is sat $\Leftarrow$ $\tau(\varphi)$ is sat)** Suppose that $\tau(\varphi)$ is satisfiable, i.e., that there exists a model $\mathcal{M} = \langle W, R, V \rangle$ and $s \in W$ such that $\langle W, R, V \rangle, s \models \mathit{Struct}$ and $\langle W, R, V \rangle, s \models \mathsf{Tr}(\varphi)$. We define $\mathcal{M}' = \langle W', R', V', \emptyset \rangle$, where

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

Thus $\mathcal{M}'$ is extracted from the translated model $\mathcal{M}$, and because $\langle W, R, V \rangle, s \models \mathsf{Tr}(\varphi)$, there is $w' \in W'$ such that $(s, w') \in R$ and $\langle W, R, V \rangle, w' \models (\varphi)'$. We will prove:

$$\langle W', R', V', M' \rangle, v' \models \psi \;\text{ iff }\; \langle W, R_{M'}, V \rangle, v' \models (\psi)' \tag{4.12}$$

where $R_{M'} = (R \backslash S^{-1}) \cup S$ with $S = \{(m', s) \mid m' \in M'\}$. In particular, when $M' = \emptyset$ we have that $\langle W', R', V', \emptyset \rangle, w' \models \varphi$ iff $\langle W, R, V \rangle, w' \models (\varphi)'$.

We now prove (4.12) by structural induction on ψ. Boolean and modal cases are easy, and it is also the case for ⓚ. Let us prove the interesting case:

$\psi = ⓡ\phi$:

($\Leftarrow$) Suppose that $\langle W, R_{M'}, V \rangle, v' \models (ⓡ\phi)'$. By definition of $(\ )'$ and the semantics we have the conjunction of the following formulas:

$$\begin{array}{ll} (1) & \langle W, R_{M'}, V \rangle, v' \models \neg \Diamond s \to \langle \mathsf{gsw} \rangle (\Diamond s \wedge (\phi)') \\ (2) & \langle W, R_{M'}, V \rangle, v' \models \Diamond s \to (\phi)' \end{array}$$

We have to prove that $\langle W', R', V', M' \rangle, v' \models ⓡ\phi$. We will prove it in two parts, first assuming (1) and then assuming (2).

First, assume (1). Now suppose $\langle W, R_{M'}, V \rangle, v' \models \neg \Diamond s$, i.e, that $(v', s) \notin R_{M'}$. Then we have that $\langle W, R_{M'}, V \rangle, v' \models \langle \mathsf{gsw} \rangle (\Diamond s \wedge (\phi)')$. By the semantics and our last assumption we have $\langle W, (R_{M'})^*_{v's}, V \rangle, v' \models \Diamond s \wedge (\phi)'$. The first conjunct is trivial because $(v', s) \in (R_{M'})^*_{v's}$ and by Proposition 4.4.9, so by applying induction hypothesis on the second conjunct, and because we know $v' \in M'$ by definition of $R_{M'}$, we have $\langle W', R', V', M' \cup \{v'\} \rangle, v' \models \phi$. Hence by the semantics we have $\langle W', R', V', M' \rangle, v' \models ⓡ\phi$.

Finally, assume (2). Now suppose $\langle W, R_{M'}, V \rangle, v' \models \Diamond s$, i.e, that $(v', s) \in R_{M'}$, which in turn also tell us that $(s, v') \notin R_{M'}$, by definition of $R_{M'}$ and Proposition 4.4.9, and therefore v' is in the memory. Then we have that $\langle W, R_{M'}, V \rangle, v' \models (\phi)'$. By induction hypothesis and because we know $v' \in M'$, we have $\langle W', R', V', M' \cup \{v'\} \rangle, v' \models \phi$, which is equivalent to $\langle W', R', V', M' \rangle, v' \models ⓡ\phi$ by the semantics.

($\Rightarrow$) Suppose $\langle W', R', V', M' \rangle, v' \models ⓡ\phi$. We have to prove $\langle W, R_{M'}, V \rangle, v' \models (ⓡ\phi)'$, i.e, we have to prove $\langle W, R_{M'}, V \rangle, v' \models \neg \Diamond s \to \langle \mathsf{gsw} \rangle (\Diamond s \wedge (\phi)')$ and $\langle W, R_{M'}, V \rangle, v' \models \Diamond s \to (\phi)'$. It is easy to prove each part separately (steps are similar to that of the ($\Leftarrow$) direction).

(φ is sat $\Rightarrow$ $\tau(\varphi)$ is sat) Suppose that φ is satisfiable, i.e., that 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 $\mathcal{M}' = \langle W', R', V'\rangle$ as follows:

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

By construction of $\mathcal{M}'$ it is easy to check that $\langle W', R', V'\rangle, s \models \mathit{Struct}$, so it only remains to see that $\langle W', R', V'\rangle, s \models \mathsf{Tr}(\varphi)$. We can verify that

$$\langle W, R, V, M\rangle, w \models \varphi \;\text{ iff }\; \langle W', R'_M, V'\rangle, s \models \mathsf{Tr}(\varphi) \tag{4.13}$$

where $R'_M = (R' \backslash S^{-1}) \cup S$ with $S = \{(m, s) \mid m \in M\}$.

We prove (4.13) by structural induction. The base cases, p and Ⓚ, are trivial; the Boolean cases, $\neg\phi$ and $\phi \wedge \chi$, follow by induction hypothesis; and the modal case, $\Diamond\phi$, is easy to prove. As for the Ⓡϕ case, note that if $\langle W, R, V, M\rangle, w \models$ Ⓡϕ, we can swap the edge (s, w) to simulate the storing of w in the memory (if $(s, w) \notin R'$ and $(w, s) \in R'$ means that $w \in M$) and continue by evaluating the rest of the translation $(\ )'$. □

From the previous theorem, we immediately get:

Theorem 4.4.12. *The satisfiability problem of $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$ is undecidable.*

To sum up, in this chapter we showed that all of the relation-changing logics are undecidable. To prove that the satisfiability problem of $\mathcal{ML}(\blacklozenge)$ is undecidable, for $\blacklozenge \in \{\langle\mathsf{sb}\rangle, \langle\mathsf{gsb}\rangle, \langle\mathsf{br}\rangle, \langle\mathsf{gbr}\rangle, \langle\mathsf{sw}\rangle, \langle\mathsf{gsw}\rangle\}$, we made a reduction from the satisfiability problem of $\mathcal{ML}$(Ⓡ, Ⓚ) to the one of $\mathcal{ML}(\blacklozenge)$. To make the reductions we relied on the spy point technique we presented in Chapter 3. The main idea was to simulate the behavior of $\mathcal{ML}$(Ⓡ, Ⓚ) without having an external memory. We simulated the ability to store states in a memory with the ability to change the accessibility relation of a model, and checking for membership in the memory was simulated by checking for changes in the accessibility relation.

With these results we have determined the borderline between decidability and undecidability for relation-changing logics, completing in this way the panorama we presented in Table 2.1 at the end of Chapter 2.

CHAPTER 5

Conclusions

In this thesis we have investigated the finite model property and the satisfiability problem of a family of logics that we call *relation-changing logics*, which extend the classical modal logic. We studied, in detail, modal operators that can delete, add, and swap an edge in the accessibility relation of a model during evaluation. It is now time to make a balance of what we have learned so far.

5.1 The Things We've Learned

We started this thesis by introducing modal and dynamic logics. In Chapter 1 we gave a brief overview of the recent historical development of modal logic, strictly understood as the logic of necessity and possibility, and particularly the historical development of systems of modal logic both syntactically and semantically, from Lewis' pioneering work starting in 1918 to Kripke's work in the early 1960's. We also introduced modal logic from a more contemporary perspective by introducing the basic modal language along with a discussion of its computational properties. We then introduced dynamic logics observing that, in this thesis, the term "dynamic" refers to logics that can change the underlying structure, and we introduced dynamic epistemic logic as an example. We discussed public announcement logic ($\mathcal{PAL}$) and showed that it can model the evolution of the knowledge of epistemic agents via updates to the model representing their epistemic state. $\mathcal{PAL}$ is not the only epistemic logic that can model dynamic behavior. For example, global and local graph modifiers are introduced in [ABdCH09] and include operators that can add and delete states and edges of the model. Another example is *arrow update logic* [KR11a, KR11b] which has operators that can delete edges depending on their pre and postconditions. All these examples show different forms of dynamics related to the topic of this thesis. We noticed that even though dynamic operators have already been investigated in the past, in this thesis we would focus on the behavior of those which can modify the accessibility relation.

In Chapter 2 we presented the family of relation-changing logics. We introduced the formal syntax and semantics of the sabotage, bridge, and swap logics, each of them in a local and global version. Each language is a syntactic extension of the basic modal logic with a dynamic operator. Semantics is based on Kripke models and model variants, which are operations that modify the model capturing

the behavior of the new syntactic operators. We observed that, even though the semantics conditions look innocent, the behavior of the operators is quite complex. Indeed, we presented some examples that describe complex properties in the models, and we also pointed out to some references in the bibliography which show evidence on their high expressive power.

In Chapter 3 we showed that relation-changing logics are more expressive that the basic modal logic by proving they lack the finite model property. For each of the sabotage, bridge, and swap logics, both with local and global effects, we enforced a formula that can only be satisfied in an infinite model. To achieve this we used a spy point technique, i.e, we forced a state of the model to be linked to every other state of the model, and we used it to describe a serial, irreflexive, and transitive set of states, implying that the models obtained are infinite. We observed that the ability to enforce infinite models shows that relation-changing logics are quite expressive, and that they could easily cross the border of decidability.

In Chapter 4 we demonstrated that the satisfiability problem for relation-changing logics is undecidable. We attempted to determine undecidability in two different ways: by encoding the $\mathbb{N} \times \mathbb{N}$ tiling problem, and by encoding the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$. In this thesis, we decided to present the latter option. We reduced the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$ to the one of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, $\mathcal{ML}(\langle \mathsf{br} \rangle)$, $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$, $\mathcal{ML}(\langle \mathsf{sw} \rangle)$, and $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$. All the reductions followed the same format: we presented a computable function which is divided in two parts. On the one hand, it defines a formula that enforces some constraints in the shape of the models using the spy point technique, to translate $\mathcal{ML}(ⓡ, ⓚ)$-models to standard Kripke models. On the other hand, it defines a translation from $\mathcal{ML}(ⓡ, ⓚ)$-formulas to formulas of each relation-changing logic, to simulate the storing of a point in a memory. In this way, we obtained undecidability results for the sabotage, bridge, and swap logics, both in their local and global versions. The encodings obtained are tricky, some being more involved than others.

Our work in this thesis ended in Chapter 4, but of course, determining the decidability or undecidability of a logic is just a first step. If it is decidable, one is interested in its complexity and in a practical algorithm. (Unfortunately, this is not the case for relation-changing logics.) If it is undecidable, one may be theoretically interested in its degree of undecidability in terms of recursion-theoretic hierarchies. On the other hand, one may try to isolate a decidable fragment that is still expressive enough to describe an interesting portion of the problems the logic was design for. Although investigating the degree of undecidability of a logic is interesting, especially if the undecidability results are obtained via tiling problems, for relation-changing logics it would be interesting to isolate decidable fragments. In fact, attempts to define new relation-changing logics that are decidable have been investigated, e.g., in [AvDFS14].

5.2 It's Been an Exciting Adventure

Working on this thesis has been an amazing adventure. In this section I want to explain how I ended up working on the topic of this thesis, sharing some of my experiences along the way.

It all started at the Latin American Congress of Mathematicians (CLAM) in August, 2012. The conference took place at the Faculty of Mathematics, Astronomy and Physics (FaMAF) of the University of Córdoba (UNC), in Argentina, and was jointly organized with the annual meeting of the Argentine Mathematical Union

(UMA). I went traveling to Córdoba, along with some students of Mathematics from Río Cuarto, to attend those conferences. At the CLAM, I attended a talk titled "Logics to Describe and Change" given by Carlos Areces. That was when I first heard about logics that could change the model, and to be honest, Carlos had really caught my attention. I still remember how amazed I was with these logics. In the talk, he presented some results on computational complexity and expressive power of swap logic and memory logics, followed by a discussion of some open problems. I wanted to know more about the topic so I asked Carlos if he could share the slides of his talk with me, and he agreed and gave me his e-mail address so that I could ask for them.

During the last day of the conference I learned about a computational complexity course that was going to take place at FaMAF during the next semester, and I was really interested in it. Although that was the first time I visited FaMAF, I already knew some people there from other conferences I had attended in previous years, so I looked for them and talked to them (here I could name Franco Luque and Ezequiel Orbe). They told me that the course was going to be taught by Guillaume Hoffmann, but that day he wasn't at FaMAF so I was given his e-mail address to contact him. I returned to Río Cuarto, wrote to him, and a few weeks later I was taking a computational complexity course at FaMAF.

Among the attendees of the course was Carlos Areces, because it turned out that Guillaume Hoffmann was a former PhD student of his. Raul Fervari, with whom I started studying my Master's degree in Computer Science in Río Cuarto a long time ago, showed up later on, because he had been on a trip to Poland for a few months. So there they were, Guillaume Hoffman giving his course, and Carlos Areces and Raul Fervari attending it. At that time, Raul was a PhD student under Carlos' supervision, and it turned out that he was working with relation-changing logics in his PhD thesis. As I already knew Raul, and because I remembered how much I liked the topic of the talk Carlos gave at the CLAM, I asked Raul if we could talk to Carlos to find out if I could work in my Master's thesis on a topic related to modal logics. So I asked Carlos and he agreed. The next week, we had a meeting with Carlos, and even though I wasn't sure what I was looking for, he knew how to guide me to find a topic that interested me. Then we decided to work on the satisfiability problem of the local version of the sabotage and bridge logics. I suppose that was when this thesis started.

I was traveling to Córdoba once a week, usually on Thursdays. In the mornings, we were gathering together with Carlos and Raul to talk about modal logics, relation-changing logics, and the techniques to prove (un)decidability results. In the afternoons, we were attending the computational complexity course with Guillaume. This went on for almost three months, throughout the course, which was completed by the end of the year. During this period, I had an incredible time learning about modal logics and computational complexity. It was an exciting and very enriching experience.

Once the course had finished, we stayed in contact with Carlos, Raul, and Guillaume via e-mail. During this time, I worked mostly with Raul and Guillaume exchanging countless e-mails and chats. They were able to read and bring comments on any scribble I made, no matter how unpolished it was. Their feedback was extremely helpful in gaining a good understanding on the behavior of relation-changing logics which ultimately helped me write formulas to enforce infinite models and to make the reductions in the undecidability proofs. Of course, we continued to have occasional meetings with Carlos, Raul, and Guillaume in Córdoba, to polish the results and bring all the pieces together. By mid 2013 the results of this thesis

were ready.

Throughout this adventure I have read and learned a lot about logics, computability and computational complexity, having a lot of fun in the process. Although it is important to learn new things and obtain new results, I believe that it is even more important to *share them*. Share your ideas, obstacles, and solutions. This is why I submitted an abstract to give a talk at the annual meeting of the UMA, which took place at the University of Rosario, Argentina, in September, 2013. So there I was, one year after I had attended the talk that Carlos gave at the CLAM, giving my own talk on the undecidability of relation-changing logics. I had the opportunity to share my work in front of a diverse audience, and I really enjoyed it.

After the talk in Rosario, the writing process of the thesis began, and a couple of months later, the first draft was ready. Today, I can say that it is now completed.

All in all, working on this thesis has been a challenging, exciting, and rewarding experience. I learned fascinating things and met interesting people, and also gained experience in research on a topic that really interests me. This is the end of this adventure, but I think it is also the beginning of an even bigger one.

APPENDIX A

Basics of the Theory of Computation

We briefly introduce some basic concepts from computability and complexity theory that are used in this thesis. More information can be found in classical textbooks such as [LP97, Sip12, Pap94, AB09].

Computability Theory

To prove that some problem is *not* computable we need a robust mathematical model of computability. One of the most widely used models is the *Turing machine*. Let f be a function, and suppose we have fixed some convention about how the elements of the domain and range of the function are to be represented. Then f is *computable* (or *recursive*) if there is a Turing machine that when given (the representation of) an item x in the domain of f will halt after finitely many steps, leaving on the tape (the representation of) $f(x)$.

We can use Turing machines to provide yes/no answers to problems. Many logical problems (for example, if some formula φ is satisfiable or not) are of this type. Suppose we have fixed the alphabet of a Turing machine, and have decided how we are going to represent the problems we are interested in. (For example, in Chapter 6 of [BdRV01], it is shown how to encode modal formulas and models as strings of 0s and 1s.) Given the encodings, some strings over the alphabet represent problem instances for which the answer is *yes*, while others represent problem instances for which the answer is *no*. A problem is *computable* (or *recursive*, or *decidable*) if there is a Turing machine which when given (the representation of) any instance of the problem, halts after finitely many steps leaving (the representation of) the correct answer on the tape.

Definition A.1 (**Church's Thesis**)**.** A problem is decidable precisely when it can be solved using a Turing machine.

Church's Thesis can be viewed as saying that computable problems are those which can be solved by writing a program in your favorite programming language when no limitations are placed on memory or execution time. In essence, Church's Thesis affirms that we *do* have a robust model of computation.

The most important benefit of having a robust definition of computability is that it gives us a way of proving that some problem is *undecidable*. We can use the

$$\begin{array}{rcl} \mathsf{P} & = & \cup_{c\geq 1}\mathsf{DTIME}(n^c) \\ \mathsf{NP} & = & \cup_{c\geq 1}\mathsf{NDTIME}(n^c) \\ \mathsf{PSPACE} & = & \cup_{c\geq 1}\mathsf{PSPACE}(n^c) \\ \mathsf{EXPTIME} & = & \cup_{c\geq 1}\mathsf{DTIME}(2^{n^c}) \\ \mathsf{NEXPTIME} & = & \cup_{c\geq 1}\mathsf{NDTIME}(2^{n^c}) \end{array}$$

Table A.1: Complexity classes.

definition of a Turing machine to prove undecidability, but it is usually easier to show that problems are undecidable via reductions:

Definition A.2. Let Σ be an alphabet and let $L_1, L_2 \subseteq \Sigma^*$ be problems (we adapt an abstract view of problems here). A *reduction from L_1 to L_2* is a computable function $\tau : \Sigma^* \to \Sigma^*$ such that $x \in L_1$ if and only if $\tau(x) \in L_2$.

The following theorem has been our main tool for proving undecidability.

Theorem A.3. *Let $L_1, L_2 \subseteq \Sigma^*$ be problems, and τ a reduction from L_1 to L_2. If L_1 is undecidable, then so is L_2.*

Complexity Theory

Complexity theory studies the computational resources required to solve (decidable) problems. The two main resources studied are *time* (the number of computation steps required) and *space* (the amount of memory required). Both time and space required are measured as functions of the length of the input.

Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A problem L is in $\mathsf{DTIME}(T(n))$ (in $\mathsf{PSPACE}(T(n))$) if and only if there is a Turing machine that runs in time (in space) $c \cdot T(n)$ for some constant $c > 0$ and decides L. Similarly, we define the class $\mathsf{NDTIME}(T(n))$ ($\mathsf{NPSPACE}(T(n))$) using non-deterministic Turing machines instead of deterministic ones.

These notions can be used to define the complexity classes we mentioned in this thesis, which are listed in Table A.1. Each of these classes is contained in the classes appearing below it in the list. Although a problem that belongs to any of these classes is decidable, P (the class at the bottom of this putative hierarchy) is widely taken to be the class of problems that are *tractable*, or *efficiently solvable*.

In this thesis we mentioned often the class PSPACE, which is the complexity class of most relevance to modal logic, and particularly, we characterized the complexity of the model checking problems of our logics of interest as being PSPACE-complete. So we end this appendix defining the concepts of hardness and completeness, but in order to do that, we first introduce the notion of polynomial time reductions.

A *polynomial time reduction* is a reduction in the sense of Definition A.2, but with τ being a *polynomial time* computable function. It follows that if L_2 is solvable in polynomial time (that is, if L_2 is tractable), then so is L_1. This is why we say in this case that L_2 is *at least as hard* as L_1, and this observation leads us to our last fundamental definition:

Definition A.4 (Hardness and Completeness). Let C be a class of problems. A problem L is C-hard (with respect to polynomial time reductions) if every problem in C is polynomial time reducible to L; L is C-complete if it is C-hard and moreover $L \in \mathsf{C}$. That is, the C-complete problems are the hardest problems in C.

Bibliography

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. Cited on page 65.

[ABdCH09] G. Aucher, P. Balbiani, L. Fariñas del Cerro, and A. Herzig. Global and local graph modifiers. *Electronic Notes in Theoretical Computer Science*, 231:293–307, 2009. Cited on page 61.

[AFFM08] 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, 2008. Cited on page 37.

[AFFM11] 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 37.

[AFGM09] 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*, pages 47–61, 2009. Cited on page 37.

[AFH12] C. Areces, R. Fervari, and G. Hoffmann. Moving Arrows and Four Model Checking Results. In *Logic, Language, Information and Computation*, volume 7456 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2012. Cited on pages 21, 22, and 24.

[AFH13] C. Areces, R. Fervari, and G. Hoffmann. Tableaux for Relation-Changing Modal Logics. In *Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 263–278, 2013. Cited on page 22.

[AFH14] C. Areces, R. Fervari, and G. Hoffmann. Swap Logic. *Logic Journal of the IGPL*, 22(2):309–332, 2014. Cited on pages 21, 22, 31, and 53.

[Are07] C. Areces. Hybrid Logics: The Old and the New. In *Proceedings of LogKCA-07*, pages 15–29, 2007. Cited on page 36.

[AvDFS14] C. Areces, H. van Ditmarsch, R. Fervari, and F. Schwarzentruber. Logics with Copy and Remove. In *Proceedings of the 21st International Workshop on Logic, Language, Information and Computation*, WOLLIC '14, Valparaíso, Chile, 2014. Cited on page 62.

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001. Cited on pages 5, 23, 24, and 65.

[Ber66] R. Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, Technical Report 66, 1966. Cited on page 35.

[BGG01] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Universitext. Springer, 2001. Cited on page 35.

[Bir33] G. Birkhoff. On the combination of subalgebras. *Proceedings of the Cambridge Philosophical Society*, 29:441–464, 1933. Cited on page 3.

[Bir35] G. Birkhoff. On the structure of abstract algebras. *Proceedings of the Cambridge Philosophical Society*, 31:433–454, 1935. Cited on page 3.

[Boo95] G. Boolos. *The Logic of Provability*. Cambridge University Press, 1995. Cited on page 3.

[BS84] R. Bull and K. Segerberg. Basic Modal Logic. In *Extensions of Classical Logic*, volume 2 of *Handbook of Philosophical Logic*, pages 1–88. Springer, 1984. Cited on page 4.

[BS95] P. Blackburn and J. Seligman. Hybrid Languages. *Journal of Logic, Language and Information*, 4(3):251–272, 1995. Cited on page 25.

[BvB06] P. Blackburn and J. van Benthem. Modal Logic: A Semantic Perspective. In *Handbook of Modal Logic*, pages 1–84. Elsevier, 2006. Cited on page 5.

[Car46] R. Carnap. Modalities and Quantification. *Journal of Symbolic Logic*, 11:33–64, 1946. Cited on page 4.

[Car47] R. Carnap. *Meaning and Necessity*. University of Chicago Press, Chicago, 1947. Cited on page 4.

[EFT96] H-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Undergraduate Texts in Mathematics. Springer, 2nd edition, 1996. Cited on page 7.

[End01] H. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2nd edition, 2001. Cited on page 7.

[Fer12] R. Fervari. The Expressive Power of Swap Logic. In *Proceedings of ESSLLI 2012 Student Session*, Opole, Poland, 2012. Cited on pages 21 and 22.

[Fer14a] R. Fervari. *Relation-Changing Modal Logics*. PhD thesis, Facultad de Matemática, Astronomía y Física, Universidad Nacional de Córdoba, Argentina, 2014. Cited on pages 21, 22, 24, and 37.

[Fer14b] R. Fervari. The Impact of Including Model Update Operators in Modal Logics. In *ESSLLI 2012 and ESSLLI 2013 Student Sessions, Selected Papers*, 2014. Cited on page 22.

[FHMV04] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. The MIT Press, 2004. Cited on page 10.

[FL79] M. Fischer and R. Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979. Cited on page 8.

[GKV97] 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 8.

[Göd33] K. Gödel. Eine interpretation des intuitionistischen aussagenkalküls. *Ergebnisseeines mathematischen Kolloquiums*, 4:39–40, 1933. English translation in Solomon Feferman *et al.*, editors, *Kurt Gödel, Collected Works, Volume 1*, p. 217. Oxford University Press, 1986. Cited on page 2.

[Gol06] R. Goldblatt. Mathematical Modal Logic: A View of its Evolution. In *Logic and the Modalities in the Twentieth Century*, volume 7 of *Handbook of the History of Logic*, pages 1–98. Elsevier, 2006. Cited on page 1.

[Har58] R. Harrop. On the existence of finite models and decision procedures for propositional calculi. *Proceedings of the Cambridge Philosophical Society*, 54:1–13, 1958. Cited on page 3.

[Har85] D. Harel. Recurring dominoes: Making the highly undecidable highly understandable. *Annals of Discrete Mathematics*, 24:51–72, 1985. Cited on page 35.

[Har86] D. Harel. Effective Transformations on Infinite Trees, with Applications to High Undecidability, Dominoes and Fairness. *Journal of the ACM*, 33:224–248, 1986. Cited on page 35.

[Har00] D. Harel. *Dynamic Logic*. Foundations of Computing. The MIT Press, 2000. Cited on page 8.

[Hin61] J. Hintikka. Modality and Quantification. *Theoria*, 27:119–128, 1961. Cited on page 5.

[Hin62] J. Hintikka. *Knowledge and Belief. An Introduction to the Logic of the Two Notions*. Cornell University Press, 1962. Cited on page 9.

[JT51] B. Jónsson and A. Tarski. Boolean Algebras with Operators. Part I. *American Journal of Mathematics*, 73:891–939, 1951. Cited on page 4.

[JT52] B. Jónsson and A. Tarski. Boolean Algebras with Operators. Part II. *American Journal of Mathematics*, 74:127–162, 1952. Cited on page 4.

[Kan57] S. Kanger. *Provability in Logic*. Stockholm Studies in Philosophy I, 1957. Cited on page 5.

[KR11a] B. Kooi and B. Renne. Arrow Update Logic. *Review of Symbolic Logic*, 4(4):536–559, 2011. Cited on page 61.

[KR11b] B. Kooi and B. Renne. Generalized arrow update logic. In *Theoretical Aspects of Rationality and Knowledge, Proceedings of the Thirteenth Conference*, pages 205–211, 2011. Cited on page 61.

[Kri59] S. Kripke. A Completeness Theorem in Modal Logic. *Journal of Symbolic Logic*, 24:1–14, 1959. Cited on page 5.

[Kri63] 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 pages 5 and 10.

[Lad77] R. Ladner. The Computational Complexity of Provability in Systems of Modal Propositional Logic. *SIAM Journal on Computing*, 6(3):467–480, 1977. Cited on page 8.

[Lew12] C. I. Lewis. Implication and the Algebra of Logic. *Mind*, 21(84):522–531, 1912. Cited on page 1.

[Lew18] C. I. Lewis. *A Survey of Symbolic Logic*. University of California Press, 1918. Cited on page 2.

[LL32] C. I. Lewis and C. H. Langford. *Symbolic Logic*. Dover, New York, 1932. Cited on page 2.

[LP97] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 2nd edition, 1997. Cited on pages 35 and 65.

[LR03a] C. Löding and P. Rohde. Model Checking and Satisfiability for Sabotage Modal Logic. In *Foundations of Software Technology and Theoretical Computer Science*, volume 2914 of *Lecture Notes in Computer Science*, pages 302–313, 2003. Cited on pages 22 and 43.

[LR03b] C. Löding and P. Rohde. Solving the sabotage game is PSPACE-hard. In *Mathematical Foundations of Computer Science*, volume 2747 of *Lecture Notes in Computer Science*, pages 531–540. Springer, 2003. Cited on page 22.

[Lut06] C. Lutz. Complexity and Succinctness of Public Announcement Logic. In *Proceedings of the 5th International joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 137–143, New York, NY, USA, 2006. Cited on page 14.

[McK41] J. C. C. McKinsey. A Solution of the Decision Problem for the Lewis Systems S2 and S4, with an Application to Topology. *Journal of Symbolic Logic*, 6:117–134, 1941. Cited on page 3.

[Mer09] S. Mera. *Modal Memory Logics*. PhD thesis, Université Henri Poincaré, France and Universidad de Buenos Aires, Argentina, 2009. Cited on pages 36 and 37.

[Mon60] R. Montague. Logical Necessity, Physical Necessity, Ethics, and Quantifiers. *Inquiry*, 3:259–269, 1960. Cited on page 5.

[MT48] J. C. C. McKinsey and A. Tarski. Some Theorems About the Sentential Calculi of Lewis and Heyting. *Journal of Symbolic Logic*, 13:1–15, 1948. Cited on page 3.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. Cited on page 65.

[Pla07] J. Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007. First published in 1989. Cited on page 11.

[Pri62] A. N. Prior. Possible Worlds. *Philosophical Quarterly*, 12:36–43, 1962. Cited on page 5.

[Rob71] R. M. Robinson. Undecidability and Nonperiodicity for Tilings of the Plane. *Inventiones mathematicae*, 12:177–209, 1971. Cited on page 35.

[Roh06] P. Rohde. *On games and logics over dynamically changing structures*. PhD thesis, RWTH Aachen University, Germany, 2006. Cited on pages 19 and 20.

[Sco62] D. Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:377, 1962. Cited on page 8.

[Sip12] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition, 2012. Cited on page 65.

[Smu95] R. Smullyan. *First-Order Logic*. Dover Books on Mathematics. Dover Publications, 1995. Cited on page 7.

[Var96] M. Vardi. Why is Modal Logic So Robustly Decidable? In *Descriptive Complexity and Finite Models*, pages 149–184, 1996. Cited on page 8.

[vB05] J. van Benthem. An Essay on Sabotage and Obstruction. In *Mechanizing Mathematical Reasoning*, pages 268–276, 2005. Cited on page 18.

[vDvdHI12] H. van Ditmarsch, W. van der Hoek, and P. Iliev. Everything is Knowable - How to Get to Know Whether a Proposition is True. *Theoria*, 78(2):93–114, 2012. Cited on page 12.

[vDvdHK07] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Synthese Library. Springer, 2007. Cited on page 11.

[vW51] G. H. von Wright. *An Essay in Modal Logic*. Amsterdam, North-Holland Pub. Co., 1951. Cited on pages 3 and 9.

[Wan61] H. Wang. Proving Theorems by Pattern Recognition II. *The Bell System Technical Journal*, 40(1):1–41, 1961. Cited on page 35.