The Impact of Including Model Update Operators in Modal Logics

Raul Fervari

FaMAF, Universidad Nacional de Córdoba & CONICET, Argentina
fervari@famaf.unc.edu.ar

Abstract. In this paper we discuss ideas about *dynamic modal logics*. Modal logics are appropriate to describe properties of relational structures, and several operators have been already introduced to describe dynamic properties of such structures. However, we are interested in those operators which can modify models during the evaluation of a formula. First, we introduce different dynamic operators to clarify which of them are interesting for us. Then we focus on operators which modify the accessibility relation of relational models, and we show some expressivity results.

Keywords: modal logics, model updates, bisimulation, complexity.

1 What kind of dynamic logics?

Modal logics [8,9] extend classical logics with operators that represent the modal character of some situation, for instance, necessity, possibility, knowledge, belief or permissions, just to name a few. In particular, the Basic Modal Logic ($\mathcal{ML}$) is an extension of propositional logic with a new operator which can describe the structural properties of a relational model. Formally:

Definition 1 (Syntax). *Let* PROP *be an infinite, countable set of propositional symbols. The set* FORM *of $\mathcal{ML}$ formulas over* PROP *is defined as:*

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

where $p \in$ PROP and $\varphi, \psi \in$ FORM. We use $\Box\varphi$ as a shorthand for $\neg\Diamond\neg\varphi$, while $\top$ and $\varphi \vee \psi$ are defined as usual.

Definition 2 (Semantics). *A model $\mathcal{M}$ is a triple $\mathcal{M} = \langle W, R, V\rangle$, where W is a non-empty set; $R \subseteq W \times W$ is the accessibility relation; and $V : \mathsf{PROP} \to \mathcal{P}(W)$ is a valuation. Let w be a state in $\mathcal{M}$, the pair $(\mathcal{M}, w)$ is called a pointed model; we will usually drop parentheses and write $\mathcal{M}, w$. Given a pointed model $\mathcal{M}, w$ and a formula φ we say that $\mathcal{M}, w$ satisfies φ ($\mathcal{M}, w \models \varphi$) when*

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

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

As shown in Definition 2, 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. But these are *static* characteristics of the structure, i.e. properties never change after the application of certain operations. If we want to describe *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 modal logics (or actually, any logic with classical semantics) becomes less clear. We can always resort to modeling the whole space of possible evolutions of the system as a graph, but this soon becomes unwieldy. It would be more elegant to use truly dynamic modal logics with operators that can mimic the changes that the structure will undergo.

We should take some care here, because some modal operators have been devised in the past to model dynamic phenomena, but not in the sense we just mentioned. One example is *Propositional Dynamic Logic* (**PDL**) [16,12,14]. This logic is a formal system for reasoning about programs. Originally, it was designed to formalize correctness specifications and prove that those specifications correspond to a particular program. **PDL** is a modal logic that contains an infinite number of modalities $\langle\pi\rangle$, where each π corresponds to a *program*. The interpretation of $\langle\pi\rangle\varphi$ is that *"some terminating execution of π from the current state leads to a state where the property φ holds"*. The structure of a program is defined inductively from a set of basic programs $\{a, b, c, \ldots\}$ as:

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

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

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

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

Clearly, the language gives us a practical way to deal with the notion of state and change, but this is a weak notion of dynamic behaviour. Formulas do

not change the model, they only formalize program executions. We are more interested in operations that can change the model while we are evaluating a formula, i.e., *model update operators*. We will see in the next section, various concrete examples of this kind of logics.

1.1 Some examples of Dynamic Modal Logics

A typical example when we think in logics that can change the model are *Dynamic Epistemic Logics* [23]. This is a family of logics that are used to reason about knowledge and belief, with operators that let us change such knowledge or belief by communicating some information. The Epistemic Logic $\mathcal{EL}$ is an extension of Propositional Logic with the knowledge operator K_a, where a is an agent name. K_a has the same semantics of $\Box$ but in a multiagent framework: edges of models are labeled by agent names, and each K_a is interpreted on the accessibility relation labeled by a. $K_a\varphi$ is interpreted as "the agent a knows that φ is the case". This logic only represents static information, but there are different extensions to model information exchange among the agents, which involves a dynamic behaviour.

Public Announcement Logic ($\mathcal{PAL}$) was introduced in [20] (first published in 1989), as an extension of $\mathcal{EL}$ with the operator $[!\varphi]$ which communicates some common information to the agents ($\langle !\psi\rangle\varphi$ is a shorthand for $\neg[!\psi]\neg\varphi$.) The formula $[!\psi]\varphi$ is read as "after ψ is (truthfully) announced, φ is the case". The formula ψ is revealed to all the agents (the announcement is public), then φ is evaluated. Announcements are represented by removing the access to states of the model where the announced fact does not hold. We introduce the formal semantics of $\mathcal{PAL}$:

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

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

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

After making an announcement, the model is transformed to a new one and evaluation of the rest of the formula continues in the new model. Agents cannot access anymore information which contradicts the announcement: the knowledge of the agents has changed. Notice that the propositional information contained in states (the valuation) does not change. The only information affected is the knowledge that the agents have of this information.

Another family of model update logics is memory logics [4,19]. The semantics of these logics is specified on models that come equipped with a set of states called the *memory*. The simplest memory logic includes a modality ⓡ that *stores* the current point of evaluation into memory, and a modality ⓚ that verifies whether the current state of evaluation has been memorized. The memory can be seen as a special proposition symbol whose extension grows whenever the ⓡ modality

is used. In contrast with public announcements, the basic memory logic *expands* the model with an ever increasing set of memorized elements.

Definition 3 (Syntax of Memory Logics). *Given a set* PROP*, the set* FORM *of formulas of* $\mathcal{ML}(ⓡ, ⓚ)$ *over* PROP *is defined as:*

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

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

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

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

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

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

Definition 4 (Semantics of Memory Logics). *A model* $\mathcal{M} = \langle W, R, V, S\rangle$ *is an extension of an Kripke model with a memory* $S \subseteq W$*. Let* w *be a state in* $\mathcal{M}$*, we inductively define the notion of satisfiability of a formula as:*

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

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

In the definition of satisfaction, the empty initial memory ensures that no point of the model satisfies the unary predicate ⓚ unless a formula $ⓡ\varphi$ or $\langle\!\langle \mathrm{r} \rangle\!\rangle\varphi$ has previously been evaluated there. The memory logic $\mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$ does not have the $\Diamond$ operator, and its expressive power is strictly weaker than $\mathcal{ML}(ⓡ, ⓚ)$ [19,5]. However, in both cases we have a logic that is strictly more expressive than the basic modal logic $\mathcal{ML}$. We show this result with a simple example.

Example 5. Given a pointed model $\langle W, R, V, \emptyset\rangle, w$, the $\mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$-formula $\langle\!\langle \mathrm{r} \rangle\!\rangle ⓚ$ is satisfiable only if w is reflexive. The $\langle\!\langle \mathrm{r} \rangle\!\rangle$ operator remembers the current element but at the same time looks for a successor. In this case, such successor has to be in the memory, but w is the only one belonging to the memory (remember that we started with the empty memory). Then, the formula is satisfiable if only if w is his own successor. The same effect can be captured with the $\mathcal{ML}(ⓡ, ⓚ)$-formula $ⓡ\Diamond ⓚ$.

Memory logics will not only result interesting as an example of model update operator, but the logic $\mathcal{ML}(ⓡ, ⓚ)$ will be useful to prove the undecidability of some other logics. The idea is taking advantage of the model update operators to simulate the capability of memorizing elements. Then we encode the satisfiability

problem of some dynamic logics into the undecidable satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$.

Notice that all the operators introduced in this section have something in common: they all can be defined in terms of an *update function* on the models. For instance, public announcements can be represented by an update function which takes a model and some announcement, and removes all the states which do not hold such announcement. The semantics of ⓡ can be seen as a function which adds elements to the memory. We are interested in this kind of operators, that let us transform a model during the evaluation of a formula. We introduced several examples, all of them thought in a determined context. This is the main difference with the work in this article. We are not interested in the application of dynamic operators to model a particular problem, we want to explore the impact of including dynamic operators (in particular, model update operators) in modal logics. When we use this kind of operators with a particular purpose, we can ignore some details about the behaviour of the resulting logics. By investigating dynamic operators from a theoretical point of view, we can study in detail the intrinsic properties of these operators.

As we mentioned, it is possible to modify a relational model in different ways. For instance, it is possible to remove elements of the domain ($\mathcal{PAL}$), change the valuation of the model ($\mathcal{ML}(ⓡ, ⓚ)$ and $\mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$) and change the *accessibility relation*. We are particularly interested in this last family of operators, that we called *Relation-Changing Operators*. This is not a new idea: van Benthem introduced the *Sabotage Operator* which deletes arbitrarily edges in the model [22], as an example of a relation-changing operator used to model changes in the scenario of a two-player game. In the epistemic logics field, *Arrow Updates* [15] were introduced to encode dynamic epistemic logics. In [7] some relation-changing operators have been introduced as data structure modifiers. In the next section, we will introduce some other examples of relation-changing operators that will be discussed in the rest of this article.

2 Relation-Changing Operators

We will introduce some relation-changing operators that have been previously investigated in [1,2,3,11]. We will compare the results obtained by adding dynamic operators to modal logics (most of them, included in the publications we mentioned). We will consider relation-changing operators, and the examples we introduced in the previous section.

In this article, we only discuss the single addition to $\mathcal{ML}$ of the local version of some relation-changing operators, i.e., operators that perform modifications from the evaluation point. We will introduce $\langle \mathsf{sw} \rangle$, an operator that swaps around edges; $\langle \mathsf{sb} \rangle$, a local version of van Benthem's sabotage operator; and $\langle \mathsf{br} \rangle$, which adds new edges from the evaluation point to an unaccessible point. Let us formally define the syntax of these relation-changing modal logics.

Definition 6 (Syntax). *Let* PROP *be a countable, infinite set of propositional symbols. Then the set* FORM *of formulas over* PROP *is defined as:*

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

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

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

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

Definition 7 (Models and Model Variants). *A model* $\mathcal{M}$ *is a triple* $\mathcal{M} = \langle W, R, V \rangle$, *where* W *is a non-empty set whose elements are called points or states;* $R \subseteq W \times W$ *is the accessibility relation; and* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *is a valuation.*

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

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

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

Let us introduce the formal semantics of the new operators.

Definition 8 (Semantics). *Given a pointed model* $\mathcal{M}, w$ *and a formula* φ *we say that* $\mathcal{M}, w$ *satisfies* φ, *and write* $\mathcal{M}, w \models \varphi$, *when*

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

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

We will discuss the impact of considering these operators and some of those introduced in Section 1.

3 Bisimulations

Bisimulations are an important tool to investigate the expressive power of the languages. In most modal logics, bisimulations are binary relations linking elements of the domains that have the same atomic information, and preserving the relational structure of the model [8]. This is the case for $\mathcal{ML}$:

Definition 9 ($\mathcal{ML}$-Bisimulations). *Let $\mathcal{M} = \langle W, R, V\rangle$, $\mathcal{M}' = \langle W', R', V'\rangle$ be two models. A non empty relation $Z \subseteq W \times W'$ is an $\mathcal{ML}$-bisimulation if it satisfies the following conditions. If wZw' then*

(atomic harmony) *for all $p \in \mathsf{PROP}$, $w \in V(p)$ iff $w' \in V'(p)$;*
(zig) *if $(w, v) \in R$ then for some v', $(w', v') \in R'$ and vZv';*
(zag) *if $(w', v') \in R'$ then for some v, $(w, v) \in R$ and vZv'.*

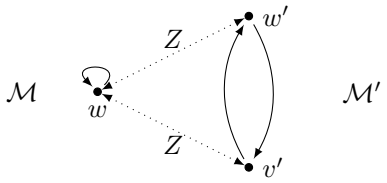
Given two pointed models $\mathcal{M}, w$ and $\mathcal{M}', w'$ we say that they are $\mathcal{ML}$-bisimilar and we write $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}} \mathcal{M}', w'$ if there is an $\mathcal{ML}$-bisimulation Z such that wZw'.

In general, when we want to express differences between two models in a particular language $\mathcal{L}$, we do it by defining an $\mathcal{L}$-formula which is satisfiable in one of them and is not satisfiable in the other. On the other hand, if we want to show that some language cannot distinguish between two models, we need specific tools to capture the expressivity of the language. As we mentioned, $\mathcal{ML}$-bisimulations relate elements in models that have the same atomic and structural information. This is exactly what we can characterize using $\mathcal{ML}$, then it looks like bisimulations are the appropriate tool to compare models. Thanks to the next theorem, we can say that if there is a bisimulation between two pointed models then they satisfy the same formulas.

Theorem 10 (Invariance for Bisimulations). *Let $\mathcal{M} = \langle W, R, V\rangle$, $\mathcal{M}' = \langle W', R', V'\rangle$ be two models, $w \in W$ and $w' \in W'$. If there is an $\mathcal{ML}$-bisimulation Z between $\mathcal{M}, w$ and $\mathcal{M}', w'$ such that wZw' then for any formula $\varphi \in \mathcal{ML}$, $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}', w' \models \varphi$.*

In a few words, the existence of an $\mathcal{ML}$-bisimulation between two models indicates that they are modally equivalent. Let us see an example.

Example 11. These two models that cannot be distinguished by any formula of $\mathcal{ML}$. Dotted lines represent the bisimulation Z.



For the Public Announcement Logic $\mathcal{PAL}$ introduced in Section 1 this is also the case: if two models are bisimilar according to Definition 9 then they satisfy the same $\mathcal{PAL}$-formulas. $\mathcal{PAL}$ has the same expressive power as $\mathcal{ML}$ [23]. The translation is not straightforward (the resulting $\mathcal{ML}$-formula can be exponentially larger than the original $\mathcal{PAL}$-formula) but it can be done via reduction axioms. However, this translation is not possible for all the dynamic logics we are discussing in this article. When we increase the expressivity of the language, the definition of bisimulation we introduced before is useless. In such cases, as we pointed in previous works, we need to include additional conditions to capture the expressivity of the logic.

Definition 12 ($\mathcal{ML}(\blacklozenge)$-Bisimulations). *Let $\mathcal{M}=\langle W, R, V\rangle$, $\mathcal{M}'=\langle W', R', V'\rangle$ be two models. A non empty relation $Z \subseteq (W \times \mathcal{P}(W^2)) \times (W' \times \mathcal{P}(W'^2))$ is a $\mathcal{ML}(\blacklozenge)$-bisimulation if it satisfies the conditions* atomic harmony, zig *and* zag *below, and the corresponding conditions for the operators that the considered logic contains. If $(w, S)Z(w', S')$ then*

(atomic harmony) *for all $p \in \mathsf{PROP}$, $w \in V(p)$ iff $w' \in V'(p)$;*
(zig) *if $(w, v) \in S$ then for some v', $(w', v') \in S'$ and $(v, S)Z(v', S')$;*
(zag) *if $(w', v') \in S'$ then for some v, $(w, v) \in S$ and $(v, S)Z(v', S')$;*
($\langle\mathsf{sb}\rangle$-zig) *if $(w, v) \in S$ then for some v', $(w', v') \in S'$ and $(v, S^{-}_{vw})Z(v' S'^{-}_{v'w'})$;*
($\langle\mathsf{sb}\rangle$-zag) *if $(w', v') \in S'$ then for some v, $(w, v) \in S$ and $(v, S^{-}_{wv})Z(v' S'^{-}_{w'v'})$;*
($\langle\mathsf{sw}\rangle$-zig) *if $(w, v) \in S$ then for some v', $(w', v') \in S'$ and $(v, S^{*}_{vw})Z(v' S'^{*}_{v'w'})$;*
($\langle\mathsf{sw}\rangle$-zag) *if $(w', v') \in S'$ then for some v, $(w, v) \in S$ and $(v, S^{*}_{vw})Z(v' S'^{*}_{v'w'})$;*
($\langle\mathsf{br}\rangle$-zig) *if $(w, v) \notin S$, there is $v' \in W'$ s.t. $(w', v') \notin S'$ and $(v, S^{+}_{wv})Z(v', S'^{+}_{w'v'})$;*
($\langle\mathsf{br}\rangle$-zag) *if $(w', v') \notin S'$, there is $v \in W$ s.t. $(w, v) \notin S$ and $(v, S^{+}_{wv})Z(v', S'^{+}_{w'v'})$.*

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

Notice that bisimulations for relation-changing modal logics relate current states and current accessibility relations of the models. Depending on which operator we are considering, different zig/zag conditions are added. Zig and zag for $\mathcal{ML}$-bisimulations are the correspondent conditions to capture $\Diamond$: they talk about the successors of the current state. Conditions for relation-changing modal logics are the same, but also keeping track of the modifications already done, and changing the relation according to the semantics of the operators. For instance, $\langle\mathsf{sb}\rangle$-zig/zag establish that there are successors of the current states that are related, and delete the edges that connect them. For $\langle\mathsf{sw}\rangle$ is the same but swapping edges instead deleting. Conditions for $\langle\mathsf{br}\rangle$ require that there exist unreachable points from the current states, and put edges to them in the accessibility relation.

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

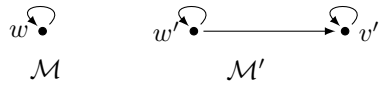
Theorem 13 (Invariance for Bisimulations). *Let $\mathcal{M} = \langle W, R, V\rangle$, $\mathcal{M}' = \langle W', R', V'\rangle$ be two models, $w \in W$, $w' \in W'$, and let $S \subseteq W^2, S' \subseteq W'^2$. If there is a $\mathcal{ML}(\blacklozenge)$-bisimulation Z between $\mathcal{M}, w$ and $\mathcal{M}', w'$ such that $(w, S)Z(w', S')$ then for any formula $\varphi \in \mathcal{ML}(\blacklozenge)$, $\langle W, S, V\rangle, w \models \varphi$ iff $\langle W', S', V'\rangle, w' \models \varphi$.*

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

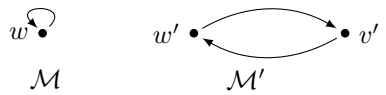
- $\boldsymbol{\varphi = \Diamond\psi}$: Suppose $\langle W, S, V\rangle, w \models \Diamond\psi$. Then there is v in W s.t. $(w, v) \in S$ and $\langle W, S, V\rangle, v \models \psi$. By (zig) we have v' in W' such that $w'S'v'$ and $(v, S)Z(v', S')$. By I.H., $\langle W', S', V'\rangle', v' \models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

□

Example 14. The two models below are $\mathcal{ML}(\langle \mathsf{sw}\rangle)$-bisimilar. The simplest way to check this is to recast the notion of $\mathcal{ML}(\langle \mathsf{sw}\rangle)$-bisimulation as an Ehrenfeucht-Fraïssé game as the one used for $\mathcal{ML}$, but where Spoiler can also swap arrows when moving from a node to an adjacent node. It is clear that Duplicator has a winning strategy.



Example 15. There is no $\mathcal{ML}(\langle \mathsf{sw}\rangle)$-bisimulation between the models below. Indeed the formula $\langle \mathsf{sw}\rangle\Diamond\Box\bot$ is satisfied in $\mathcal{M}', w'$ and not in $\mathcal{M}, w$. Notice that the models are $\mathcal{ML}$-bisimilar.



As we have seen, the first difference between $\mathcal{PAL}$ and relation-changing modal logics is the definition of bisimulation. For $\mathcal{PAL}$ it suffices with the conditions defined for $\mathcal{ML}$, but for the relation-changing modal logics we are discussing in this article we need to define new conditions which capture the new behaviour. As we showed in [1,3,11], it is natural given that these relation-changing operators increase the expressive power of $\mathcal{ML}$.

We can use bisimulations to compare the logics among them, and also with others. Definition 16 formalizes how we compare the expressive power of two logics.

Definition 16 ($\mathcal{L} \leq \mathcal{L}'$). *We say that $\mathcal{L}'$ is* at least as expressive as *$\mathcal{L}$ (notation $\mathcal{L} \leq \mathcal{L}'$) if there is a function Tr between formulas of $\mathcal{L}$ and $\mathcal{L}'$ such that for every model $\mathcal{M}$ and every formula φ of $\mathcal{L}$ we have that*

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

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

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

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

The $\leq$ relation indicates that we can embed one language into another. To do this, we need an equivalence preserving translation from the first language to the second one. Its strict version is $<$, that indicates that the second language can express strictly more than the first one. Incomparability relation says than any of the two languages cannot be embedded in the other, i.e., they are able to say different things. These definitions will be used next, when we compare the expressive power of relation-changing modal logics of Definition 6.

The comparisons have been already investigated in [1,11], establishing that relation-changing modal logics are all incomparable among them. Some cases are easy to check, but there are others in which we need more complex structures to distinguish two languages.

Lemma 17. *For every pair of pointed models $\mathcal{M}, w$ and $\mathcal{M}', w'$ in Figure 1, and for all corresponding formulas φ of the column "Distinct by", we have $\mathcal{M}, w \not\models \varphi$ and $\mathcal{M}', w' \models \varphi$. Moreover, for all corresponding logics $\mathcal{L}$ of the column "Bisimilar for", we have that (w, R) and (w', R') are in an $\mathcal{L}$-bisimulation, where R and R' are the accessibility relations of $\mathcal{M}$ and $\mathcal{M}'$ respectively.*

Proof. We will check the conditions to show that the two models in first row are bisimilar for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ and for $\mathcal{ML}(\langle \mathsf{sw} \rangle)$. Clearly all the states agree propositionally (their valuations are empty). For zig and zag conditions, we need to check if both have bisimilar successors, which holds because there are not successors at all. The same happens with $\langle \mathsf{sb} \rangle$-zig/zag and $\langle \mathsf{sw} \rangle$-zig/zag: the lack of successors makes the conditions true. Now we can prove that $\mathcal{ML}(\langle \mathsf{br} \rangle) \not\leq \mathcal{ML}(\langle \mathsf{sb} \rangle)$ and $\mathcal{ML}(\langle \mathsf{br} \rangle) \not\leq \mathcal{ML}(\langle \mathsf{sw} \rangle)$. We have to check now that there is a $\mathcal{ML}(\langle \mathsf{br} \rangle)$-formula that distinguishes the two models. The $\mathcal{ML}(\langle \mathsf{br} \rangle)$-formula $\langle \mathsf{br} \rangle \langle \mathsf{br} \rangle \top$ holds at $\mathcal{M}', w'$ but not at $\mathcal{M}, w$. Checking $\langle \mathsf{br} \rangle$-zag, it fails starting from w' and finding two states to reach with a new edge, while starting from w we can just reach one.

The models in the second row are $\mathcal{ML}(\langle \mathsf{br} \rangle)$-bisimilar because no new edges can be added, and we checked that they are also $\mathcal{ML}(\langle \mathsf{sw} \rangle)$-bisimilar in Example 15. In the third row, the given models are bisimilar for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ because they are bisimilar for $\mathcal{ML}$ and they are acyclic. In the fourth row, both models are $\mathcal{ML}(\langle \mathsf{br} \rangle)$-bisimilar since they are infinite, hence one can add as many links as needed to points that are modally bisimilar.

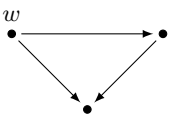
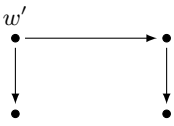
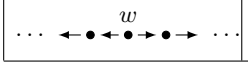
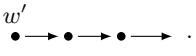
$\mathcal{M}$	$\mathcal{M}'$	Distinct by	Bisimilar for
		$\langle\mathsf{br}\rangle\langle\mathsf{br}\rangle\top$	$\mathcal{ML}(\langle\mathsf{sb}\rangle)$ $\mathcal{ML}(\langle\mathsf{sw}\rangle)$
		$\langle\mathsf{sb}\rangle\Diamond\top$	$\mathcal{ML}(\langle\mathsf{sw}\rangle)$ $\mathcal{ML}(\langle\mathsf{br}\rangle)$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

Fig. 1. Bisimilar models and distinguishing formulas.

□

Corollary 18. *For all* $\blacklozenge_1, \blacklozenge_2 \in \{\langle\mathsf{sb}\rangle, \langle\mathsf{sw}\rangle, \langle\mathsf{br}\rangle\}$ *such that* $\blacklozenge_1 \neq \blacklozenge_2$*, we have* $\mathcal{ML}(\blacklozenge_1) \neq \mathcal{ML}(\blacklozenge_2)$.

We have proved in [1,11] that adding some relation-changing operators to the basic modal logic we increase its expressive power, and according to the results we just showed, each logic allows to express different things. We have seen in this section that standard tools in modal logics such as bisimulations can be adapted for logics with relation-changing operators.

4 Computational Behaviour

When we need to choose a logic to model a particular problem, First-Order Logic $\mathcal{FOL}$ [10] comes immediately to our mind. $\mathcal{FOL}$ is a nice language, very powerful and well-known for everyone who studied mathematics and/or computer science, but it has some undesirable properties. For instance, its satisfiability problem is undecidable, and model checking is PSPACE-complete. However, $\mathcal{FOL}$ is still used because it is appropriate to model many different problems. On the other hand, there are weaker languages with a better computational behaviour that we can use, such as modal logics. For any problem that requires describing structural properties of a graph, modal logics can be a good choice. The satisfiability problem for $\mathcal{ML}$ is PSPACE-complete, and its model checking problem is in P.

These two languages have very different properties (more expressive power in $\mathcal{FOL}$, better computational properties in $\mathcal{ML}$), and each of them is still appropriate in determined situations. Let us see what happens when we add model update operators to modal logics. We will analyze if by adding the kind of operators that are appropriate to model dynamic situations, we preserve the good properties of $\mathcal{ML}$, or the increasing of the expressivity leads them closer to $\mathcal{FOL}$.

Let us start by discussing the case that, so far, resulted easier to be analyzed: the public announcement logic $\mathcal{PAL}$. We mentioned that this logic has the same expressive power than $\mathcal{ML}$ but there are certain properties that can be expressed exponentially more succinct in $\mathcal{PAL}$ than in $\mathcal{ML}$. Despite this succinctness, the computational complexity of these two logics coincides [18,13]. In this case, adding dynamic behaviour we keep the properties.

On the other hand, memory logics have a more complex behaviour. We know that $\mathcal{ML}$ is a proper fragment of $\mathcal{FOL}$ [9] with good computational properties. Memory logics are also a proper fragment of $\mathcal{FOL}$ [19], but unfortunately, the good properties of modal logics are not preserved. Adding to the language the capability of remember visited elements we move closer to $\mathcal{FOL}$ than to $\mathcal{ML}$. The satisfiability problem for $\mathcal{ML}(\langle\langle \mathsf{r} \rangle\rangle, \textcircled{\scriptsize k})$ is decidable but the one of $\mathcal{ML}(\textcircled{\scriptsize r}, \textcircled{\scriptsize k})$ is not, and its model checking problem is PSPACE-complete [4,6,19].

For relation-changing operators, we have similar results. In [3] we provided a translation from $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ to two sorted $\mathcal{FOL}$ by unraveling all the possible model transformations that can be done using $\langle \mathsf{sw} \rangle$. Sorts are convenient for such translation, but it is possible to translate it to unsorted $\mathcal{FOL}$, then we can conclude that $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ is a proper fragment of $\mathcal{FOL}$. It would be easy apply a similar argument for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ and $\mathcal{ML}(\langle \mathsf{br} \rangle)$ to prove the same result. Such as for $\mathcal{ML}(\textcircled{\scriptsize r}, \textcircled{\scriptsize k})$, even though they are proper fragments of $\mathcal{FOL}$, adding relation-changing modal operators increases the computational complexity of the logics. We have proved that the model checking task for logics of Definition 6 is PSPACE-complete [1,11] (such as for $\mathcal{FOL}$). Also, in [3] we proved in detail that the satisfiability problem for $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ is undecidable, and in [17,21] the same was showed for a global version of the sabotage operator. This results give us an idea about the computational behaviour of this kind of operators. In the next section we will use similar arguments as the used for $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ to prove that the satisfiability problem for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ is undecidable.

4.1 Undecidability of $\mathcal{ML}(\langle \mathsf{sb} \rangle)$

We will prove that the satisfiability problem for $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ is undecidable. This result has been proved together with Mauricio Martel[1] and appears in [11]. First, we provide a translation from formulas of this logic to formulas of the memory logic $\mathcal{ML}(\textcircled{\scriptsize r}, \textcircled{\scriptsize k})$. In order to simulate the behaviour of the operators $\textcircled{\scriptsize r}$ and $\textcircled{\scriptsize k}$ without having a memory in the model, we impose constraints on the models where we evaluate the translated formula. Then we prove that a $\mathcal{ML}(\textcircled{\scriptsize r}, \textcircled{\scriptsize k})$-formula is satisfiable if and only if, the translation of such formula (in addition to the constraints we define) is satisfiable.

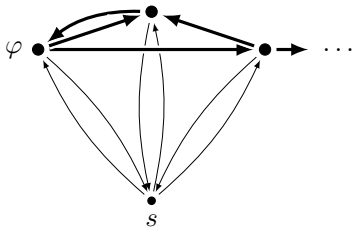
[1] Master student at Universidad Nacional de Río Cuarto, Argentina.

Definition 19. *Let $s \in \mathsf{PROP}$, we define Conds as the conjunction of the following formulas:*

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

Let us call s (for *spy point*) a node satisfying *Conds* in an arbitrary model. Then, the point s satisfies the propositional symbol s, and is related with all the states of the connected component of the model in the two directions. Formula (1) ensures that the propositional symbol s is satisfied at the evaluation point, and is not satisfied in any successor. It also says that all the successors can see an s-state. (2) ensures that all the s-states that are accessible in two steps from the evaluation point, has no successors satisfying s. (3) ensures that after deleting two edges and reaching an s-state, the property that all the successors can see an s-state is maintained. The formula (4) establishes that for all the successors, after deleting an edge an reaching an s-state, there is a successor which cannot see any s-state (it was the only successor satisfying s). (5) says that reaching some state in two steps that does not satisfy s, there is always an s-state which is reachable and has no successors satisfying s. (6) ensures that after eliminating the edge from a $\neg s$-state (which is no longer accessible from the evaluation point in two steps) to an s-state, the remaining $\neg s$-states still have an edge pointing to some state satisfying s. (7) ensures that all the states reachable in two steps (which do not satisfy s) have only one successor labeled by s. Finally, (*Spy*) establishes that states that are accessible in two steps are also accessible in one step.

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



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

Proposition 20. *Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, $w \in W$. If $\mathcal{M}, w \models Conds$, then the following properties hold:*

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

Proposition 20 enumerates the main properties of the spy point: it is the only spy point in the connected component, and each time that there is an outgoing edge to some state of the model, there is also an edge coming back.

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

Definition 21. *Let φ be an $\mathcal{ML}(ⓡ, ⓚ)$-formula that does not contain the propositional symbol s. We define $\mathsf{Tr}(\varphi) = \Diamond(\varphi)'$, where $(\ )'$ is defined as follows:*

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

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

Theorem 22. *Let φ be a formula of $\mathcal{ML}(ⓡ, ⓚ)$ that does not contain the propositional symbol s. Then, φ and $\mathsf{Tr}(\varphi) \wedge Conds$ are equisatisfiable.*

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

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

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

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

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

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

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

- $\boldsymbol{\psi = p}$: Suppose that $\langle W', R', V', M'\rangle, v \models p$. By $\models$ we have $v \in V'(p)$, and this is equivalent to $v \in V(p) \cap W'$ by definition of V'. Because $v \in V(p)$, by $\models$ we have $\langle W, R(M'), V\rangle, v \models p$, and by definition of $(\ )'$ this is equivalent to $\langle W, R(M'), V\rangle, v \models (p)'$.
- $\boldsymbol{\psi = \textcircled{k}}$: Suppose that $\langle W', R', V', M'\rangle, v \models \textcircled{k}$. By $\models$ we have $v \in M'$, and by Proposition 20 and definition of $R(M')$ we have $(v,s) \notin R(M')$ and $\langle W, R(M'), V\rangle, s \models s$. Then by $\models \langle W, R(M'), V\rangle, v \models \neg\Diamond s$, and by definition of $(\ )'$ this is equivalent to $\langle W, R(M'), V\rangle, v \models (\textcircled{k})'$.

Now we prove inductive cases.

- $\boldsymbol{\psi = \neg\phi}$: Suppose $\langle W', R', V', M'\rangle, v \models \neg\phi$. By $(\models)$, $\langle W', R', V', M'\rangle, v \not\models \phi$. By I.H., we have $\langle W, R(M'), V\rangle, v \not\models (\phi)'$, iff $\langle W, R(M'), V\rangle, v \models \neg(\phi)'$. Then, by definition of $(\ )'$, $\langle W, R(M'), V\rangle, v \models (\neg\phi)'$.
- $\boldsymbol{\psi = \phi \wedge \chi}$: Suppose $\langle W', R', V', M'\rangle, v \models \phi \wedge \chi$. By $\models$, $\langle W', R', V', M'\rangle, v \models \phi$ and $\langle W', R', V', M\rangle, v \models \chi$. By I.H. we have $\langle W, R(M'), V\rangle, v \models (\phi)'$ and $\langle W, R(M'), V\rangle, v \models (\chi)'$. Then we have $\langle W, R(M'), V\rangle, v \models (\phi)' \wedge (\chi)'$. Then by definition of $(\ )'$, $\langle W, R(M'), V\rangle, v \models (\phi \wedge \chi)'$.
- $\boldsymbol{\psi = \Diamond\phi}$: Suppose $\langle W, R(M'), V\rangle, v \models (\Diamond\phi)'$. By definition of $(\ )'$ we have $\langle W, R(M'), V\rangle, v \models \Diamond(\neg s \wedge (\phi)')$. By $\models$, there is $v' \in W$ s.t. $(v,v') \in R(M')$ and $\langle W, R(M'), V\rangle, v' \models \neg s \wedge (\phi)'$. Then we have $\langle W, R(M'), V\rangle, v' \models \neg s$ and $\langle W, R(M'), V\rangle, v' \models (\phi)'$. By I.H., $\langle W', R', V', M'\rangle, v' \models \phi$, hence by $\models$ and Proposition 20, we have $\langle W', R', V', M'\rangle, v \models \Diamond\phi$.
- $\boldsymbol{\psi = \textcircled{r}\phi}$: Suppose $\langle W, R(M'), V\rangle, v \models (\textcircled{r}\phi)'$. By definition of $(\ )'$ and $\models$,

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

We will prove each conjunct separately. First, suppose $\langle W, R(M'), V\rangle, v \models \Diamond s$. Then $(v,s) \in R(M')$ (by Proposition 20). We want to prove that

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

By assumption we know $(v,s) \in R(M')$ then $(s,v) \in R(M')$, because in $R(M')$ we always delete pairs in the two directions and by Proposition 20. Then we only need to prove that $\langle W, R(M')^{-}_{vs}, V\rangle, s \models s \wedge \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\phi)')$. It is trivial that $\langle W, (R(M'))^{-}_{vs}, V\rangle, s \models s$. Let us see $\langle W, (R(M'))^{-}_{vs}, V\rangle, s \models \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\phi)')$. Because $(s,v) \in (R(M'))^{-}_{vs}$, we only need to prove that $\langle W, R(M')^{-}_{vs,sv}, V\rangle, v \models \neg\Diamond s \wedge (\phi)'$. First conjunct is trivial because $(v,s) \notin R(M')^{-}_{vs,sv}$.
On the other hand, we know that for all t, $(t,s) \notin R(M')$ iff $(s,t) \notin R(M')$. Then by I.H., $\langle W, R(M')^{-}_{vs,sv}, V\rangle, v \models (\phi)'$ iff $\langle W', R', V', M \cup \{v\}\rangle, v \models \phi$. Hence, by $\models$ we have $\langle W', R', V', M'\rangle, v \models \textcircled{r}\phi$.

Now suppose the other case, $\langle W, R(M'), V\rangle, v \models \neg\Diamond s$. By Proposition 20, we know $(v, s) \notin R(M')$. By definition of $R(M')$, we have $(s, v) \notin R(M')$. Then $v \in M'$, and by I.H. we have $\langle W', R', V', M'\rangle, v \models \textcircled{r}\phi$.

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

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

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

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

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

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

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

From the previous theorem, we immediately get:

Theorem 23. *The satisfiability problem of $\mathcal{ML}(\langle\mathsf{sb}\rangle)$ is undecidable.*

We showed that $\mathcal{ML}(\langle\mathsf{sb}\rangle)$ behaves in the same way as $\mathcal{ML}(\langle\mathsf{sw}\rangle)$ with respect to the satisfiability problem. For the $\mathcal{ML}(\langle\mathsf{br}\rangle)$ case, similar constructions have been done in [11]. With the relation-changing operators that we introduced we can simulate memory logics operators. The idea is to use the capability of adding, swapping and deleting edges to remember points of the model. In general, by defining any operator with this ability we increase the expressivity of the language to an undecidable one.

5 Conclusions

We have discussed several cases of modal logics with operators that let us modify a model. The *Public Announcement Logic* $\mathcal{PAL}$ [20], incorporates an operator that removes all the states of the model which do not satisfy a determined formula (the *announcement*). Adding this new operator to the basic modal logic does not affect its behaviour, because public announcements can be represented

by modal formulas (which are possibly exponentially larger), and even the notion of bisimulation remains unchanged. On the other hand, *Memory Logics* [4,19] are extensions of $\mathcal{ML}$ that come equipped with operators to store states in a memory and to consult if the current state belongs to the memory. This new behaviour can be captured by adding a new propositional symbol, and changing its extension when we want to remember some state. Hence, memory logics can be seen as a model update logic with the ability of modify the valuation of the models. In this case, a new notion of bisimulation has to be defined, and the computational complexity blows up with respect to $\mathcal{ML}$.

In order to further explore the spectrum of possible modifications that can be done to a relational model, we discussed in this article *Relation-Changing Modal Logics*. Some other operators that change the accessibility relation of the models have been investigated in the past, such as van Benthem's sabotage logic [22], and some *Epistemic Logics* [7,15]. However, our goal was to investigate different relation-changing operators from a theoretical point of view to study the effects of using this kind of operators. Local Sabotage, Swap and Bridge were introduced before in [1,3,11]. In this paper we provide an analysis of their properties and compare them with other kind of model updates. We learned that it is possible to adapt the notion of bisimulation to capture their behaviour, and we obtained some experience working with relation-changing logics that can help us in the future, for instance, to find decidable fragments.

Acknowledgments: This work was partially supported by grants ANPCyT-PICT-2008-306, ANPCyT-PIC-2010-688, the FP7-PEOPLE-2011-IRSES Project "Mobility between Europe and Argentina applying Logics to Systems" (MEALS) and the Laboratoire Internationale Associé "INFINIS".

Thanks to Mauricio Martel for working with me on the undecidability results.

References

1. C. Areces, R. Fervari, and G. Hoffmann. Moving arrows and four model checking results. In L. Ong and R. Queiroz, editors, *Logic, Language, Information and Computation*, volume 7456 of *Lecture Notes in Computer Science*, pages 142–153. Springer Berlin Heidelberg, 2012.
2. C. Areces, R. Fervari, and G. Hoffmann. Tableaux for relation-changing modal logics. In P. Fontaine, C. Ringeissen, and R. Schmidt, editors, *Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 263–278. Springer, 2013.
3. C. Areces, R. Fervari, and G. Hoffmann. Swap logic. *Logic Journal of the IGPL*, 22(2):309–332, 2014.
4. C. Areces, D. Figueira, S. Figueira, and S. Mera. Expressive power and decidability for memory logics. In *Logic, Language, Information and Computation*, volume 5110 of *Lecture Notes in Computer Science*, pages 56–68. Springer Berlin / Heidelberg, 2008. Proceedings of WoLLIC 2008.
5. 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.

6. C. Areces, D. Figueira, D. Gorín, and S. Mera. Tableaux and model checking for memory logics. In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 5607 of *LNAI*, pages 47–61, Oslo, Norway, 2009. Springer Berling / Heidelberg. Proceedings of Tableaux09.
7. G. Aucher, P. Balbiani, L. Fariñas Del Cerro, and A. Herzig. Global and local graph modifiers. *Electronic Notes in Theoretical Computer Science (ENTCS), Special issue Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007)*, 231:293–307, 2009.
8. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Comp. Scie.* Cambridge University Press, 2001.
9. P. Blackburn and J. van Benthem. Modal logic: A semantic perspective. In *Handbook of Modal Logic*. Elsevier North-Holland, 2006.
10. H. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.
11. R. Fervari. *Relation-Changing Modal Logics*. PhD thesis, Facultad de Matemática Astronomía y Física, Universidad Nacional de Córdoba, Córdoba, Argentina, March 2014.
12. M. Fischer and R. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.
13. T. French, W. van der Hoek, P. Iliev, and B. Kooi. On the succinctness of some modal logics. *Artificial Intelligence*, 197:56–85, 2013.
14. D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic. Vol. II*, volume 165 of *Synthese Library*, pages 497–604. D. Reidel Publishing Co., Dordrecht, 1984. Extensions of classical logic.
15. B. Kooi and B. Renne. Arrow update logic. *Review of Symbolic Logic*, 4(4):536–559, 2011.
16. R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977.
17. C. Löding and P. Rohde. Solving the sabotage game is PSPACE-hard. In *Mathematical Foundations of Computer Science 2003*, volume 2747 of *Lecture Notes in Computer Science*, pages 531–540. Springer, Berlin, 2003.
18. C. Lutz. Complexity and succinctness of public announcement logic. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *AAMAS*, pages 137–143, Hakodate, Japan, 2006. ACM.
19. S. Mera. *Modal Memory Logics*. PhD thesis, Univ. de Buenos Aires and UFR STMIA - Ecole Doctorale IAEM Lorraine Dép. de Form. Doct. en Informat., 2009.
20. J. Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.
21. P. Rohde. *On games and logics over dynamically changing structures*. PhD thesis, RWTH Aachen, 2006.
22. J. van Benthem. An essay on sabotage and obstruction. In *Mechanizing Mathematical Reasoning*, pages 268–276, 2005.
23. H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Kluwer, 2007.