Moving Arrows and Four Model Checking Results

Carlos Areces[1,2], Raul Fervari[1] and Guillaume Hoffmann[1]

[1] FaMAF, Universidad Nacional de Córdoba, Argentina
{careces, fervari, hoffmann}@famaf.unc.edu.ar
[2] CONICET, Argentina

Abstract. We study dynamic modal operators that can change the model during the evaluation of a formula. In particular, we extend the basic modal language with modalities that are able to swap, delete or add pairs of related elements of the domain, while traversing an edge of the accessibility relation. We study these languages together with the sabotage modal logic, which can arbitrarily delete edges of the model. We define a suitable notion of bisimulation for the basic modal logic extended with each of the new dynamic operators and investigate their expressive power, showing that they are all uncomparable. We also show that the complexity of their model checking problems is PSpace-complete.

1 Introduction

Modal logics [2,4] are particularly well suited to *describe* graphs, and this is fortunate as many situations can be modeled using graphs: an algebra, a database, the execution flow of a program or, simply, the arbitrary relations between a set of elements. This explains why modal logics have been used in many, diverse fields. They offer a well balanced trade-off between expressivity and computational complexity (model checking the basic modal language $\mathcal{ML}$ is only polynomial, while its satisfiability problem is PSpace-complete). 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.

But if we want to describe and 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 modal logics (or actually, any kind of 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 structure will undergo. This is not a new idea, and a clear example of this kind of logics is the *sabotage logic* introduced by Johan van Benthem in [12].

Consider the following *sabotage game*. It is played on a graph with two players, Runner and Blocker. Runner can move on the graph from node to accessible node, starting from a designated point, and with the goal of reaching a given

final point. He should move one edge at a time. Blocker, on the other hand, can delete one edge from the graph, every time it is his turn. Of course, Runner wins if he manages to move from the origin to the final point in the graph, while Blocker wins otherwise. van Benthem discusses in [12] how to transform the sabotage game into a modal logic. van Benthem's original idea has been studied in several other works [6,10] where the sabotage operator is defined as:

$$\mathcal{M}, w \models \langle gs \rangle \varphi \quad \text{iff} \quad \text{there is a pair } (u, v) \text{ of } \mathcal{M} \text{ such that } \mathcal{M}^{-}_{\{(u,v)\}}, w \models \varphi,$$

where $\mathcal{M}^{-}_{\{(u,v)\}}$ is identical to $\mathcal{M}$ except that the edge (u, v) has been removed from the accessibility relation.

It is clear that the $\langle gs \rangle$ operator *changes* the model in which a formula is evaluated. As van Benthem puts it, $\langle gs \rangle$ is an "external" modality that takes evaluation to another model, obtained from the current one by deleting some transition. It has been proved that solving the sabotage game is PSpace-hard, while the model checking problem of the associated modal logic is PSpace-complete and the satisfiability problem is undecidable. The logic fails to have both the finite model property and the tree model property [6,10].

In this article, we will investigate various model changing operators. The first one, $\langle sw \rangle$, has the ability to swap the direction of a traversed arrow. The $\langle sw \rangle$ operator is a $\Diamond$ 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, and the pair (v, w) added to the accessibility relation. A picture will help understand the dynamics of $\langle sw \rangle$. The formula $\langle sw \rangle \Diamond \top$ is true in a model with two related states:

As we can see in the picture, evaluation starts at state w with the arrow pointing from w to v, but after evaluating the $\langle sw \rangle$ operator, it continues at state v with the arrow now pointing from v to w. We will investigate two other dynamic operators in this article. $\langle ls \rangle$, for *local sabotage*, is a $\Diamond$ operator that destroys the traversed arrow, while $\langle br \rangle$, for *bridge*, models the opposite situation: it adds an arrow to an inaccessible point of the model and moves over there.

We have chosen these model changing operators with the intention of covering a sufficiently varied sample of alternatives. The goal is to investigate whether the differences among them lead to different properties of the logics they defined, and how they vary in expressive power. 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 [10].

Summing up then, we will study and compare the expressive powers of $\mathcal{ML}(\langle sw \rangle)$, $\mathcal{ML}(\langle gs \rangle)$, $\mathcal{ML}(\langle ls \rangle)$ and $\mathcal{ML}(\langle br \rangle)$, and we provide complexity results for their model checking problems.

2 Syntax and Semantics

The syntax of the dynamic modal logics we will study is a straightforward extension of the basic modal logic (see [2]):

Definition 1 (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 $p \in \mathsf{PROP}$, $\blacklozenge \in \{\Diamond, \langle sw\rangle, \langle gs\rangle, \langle ls\rangle, \langle br\rangle\}$ *and* $\varphi, \psi \in \mathsf{FORM}$. *Other operators are defined as usual. In particular,* $\blacksquare\varphi$ *is defined as* $\neg\blacklozenge\neg\varphi$.

Formulas of the basic modal language $\mathcal{ML}$ *are those that contains only the* $\Diamond$ *operator beside the Boolean operators. We call* $\mathcal{ML}(\blacklozenge)$ *to the extension of* $\mathcal{ML}$ *allowing also the* $\blacklozenge$ *operator, for* $\blacklozenge \in \{\langle sw\rangle, \langle gs\rangle, \langle ls\rangle, \langle br\rangle\}$.

Semantically, formulas of $\mathcal{ML}(\langle sw\rangle)$, $\mathcal{ML}(\langle gs\rangle)$, $\mathcal{ML}(\langle ls\rangle)$ and $\mathcal{ML}(\langle br\rangle)$ are evaluated in standard relational models, and the meaning of all the operators of the basic modal logic is unchanged. When we evaluate formulas containing dynamic 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 article we will use wv as a shorthand for $\{(w, v)\}$ or (w, v). Context will always disambiguate the intended use.

Definition 2 (Models and Model Updates). *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:*

(swapping) $\mathcal{M}^*_{vw} = \langle W, R^*_{vw}, V\rangle$, *with* $R^*_{vw} = (R \backslash wv) \cup vw$, $wv \in R$.
(sabotaging) $\mathcal{M}^-_{wv} = \langle W, R^-_{wv}, V\rangle$, *with* $R^-_{wv} = R \backslash wv$, $wv \in R$.
(bridging) $\mathcal{M}^+_{wv} = \langle W, R^+_{wv}, V\rangle$, *with* $R^+_{wv} = R \cup wv$, $wv \in (W \times W) \backslash R$.

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.*

We are now ready to introduce the semantics.

Definition 3 (Semantics). *Given a pointed model* $\mathcal{M}, w$ *and a formula* φ *we say that* $\mathcal{M}, w$ *satisfies* φ, *and write* $\mathcal{M}, w \models \varphi$, *when*

$$\begin{array}{ll}
\mathcal{M}, w \models p & \text{iff } w \in V(p) \\
\mathcal{M}, w \models \neg\varphi & \text{iff } \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff } \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \Diamond\varphi & \text{iff for some } v \in W \text{ s.t. } wRv, \mathcal{M}, v \models \varphi \\
\mathcal{M}, w \models \langle sw\rangle\varphi & \text{iff for some } v \in W \text{ s.t. } wRv, \mathcal{M}^*_{vw}, v \models \varphi \\
\mathcal{M}, w \models \langle gs\rangle\varphi & \text{iff for some } v, u \in W, \text{ s.t. } vRu, \mathcal{M}^-_{vu}, w \models \varphi \\
\mathcal{M}, w \models \langle ls\rangle\varphi & \text{iff for some } v \in W \text{ s.t. } wRv, \mathcal{M}^-_{wv}, v \models \varphi \\
\mathcal{M}, w \models \langle br\rangle\varphi & \text{iff for some } v \in W \text{ s.t. } \neg wRv, \mathcal{M}^+_{wv}, v \models \varphi.
\end{array}$$

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

We write $\mathcal{M}, w \equiv_{\mathfrak{L}} \mathcal{N}, v$ when both models satisfy the same $\mathfrak{L}$-formulas, i.e., for all $\varphi \in \mathfrak{L}$, $\mathcal{M}, w \models \varphi$ if and only if $\mathcal{N}, v \models \varphi$. We will drop the $_{\mathfrak{L}}$ subindex when no confusion arises.

Once syntax and semantics are in place, the following result that distinguishes the dynamic logics from $\mathcal{ML}$ can be easily established. A basic result for $\mathcal{ML}$ shows that it has the *tree model property*: every satisfiable formula of $\mathcal{ML}$ can be satisfied at the root of a model where the accessibility relation defines a tree (i.e., there is a root, the relation is irreflexive, all elements different from the root can be reached from the root via the transitive closure of the accessibility relation, and no element has two different predecessors).

Theorem 4. *$\mathcal{ML}(\blacklozenge)$ does not have the tree model property, for $\blacklozenge \in \{\langle sw\rangle, \langle gs\rangle, \langle ls\rangle, \langle br\rangle\}$*

Proof. For details see the appendix. We present formulas that ensure that the accessibility relation does not define a tree. The $\langle gs\rangle$ case has already been proved in [6]. Suppose the following formulas hold at some point w:

1. $p \wedge (\bigwedge_{1 \leq i \leq 3} \Box^i \neg p) \wedge \langle sw\rangle \Diamond \Diamond p$, then w has a reflexive successor;
2. $\Diamond\Diamond\top \wedge [ls]\Box\bot$, then w is reflexive;
3. $\Diamond\Diamond\top \wedge [gs]\Box\bot$, then w is reflexive;
4. $\langle br\rangle\Box\bot$, then w and some different point v are unconnected.

In each case, the formula cannot be satisfied in a tree. □

As the four logics we introduced are conservative extensions of $\mathcal{ML}$, the formulas above show that each is strictly more expressive than $\mathcal{ML}$. A natural question is whether these dynamic logics are different from each other. We will use bisimulations to answer this question.

Because we need to keep track of the changes on the accessibility relation that the dynamic operators can introduce, we will define bisimulations as relations that link a point of evaluation together with the current accessibility relation.

Definition 5 (Bisimulations). *Given models $\mathcal{M} = \langle W, R, V\rangle$ and $\mathcal{M}' = \langle W', R', V'\rangle$, together with points $w \in W$ and $w' \in W'$ we say that they are bisimilar and write $\mathcal{M}, w \leftrightarrows \mathcal{M}', w'$ if there is a relation $Z \subseteq (W \times \mathcal{P}(W^2)) \times (W' \times \mathcal{P}(W'^2))$ such that $(w, R)Z(w', R')$ satisfying conditions from Figure 1. Which conditions have to be satisfied depends on the operators present in the language.*

If needed, we write $\leftrightarrows_{\mathfrak{L}}$ to indicate that the bisimulation corresponds to $\mathfrak{L}$.

Theorem 6 (Invariance for Dynamic Logics.). *For $\mathcal{ML}(\blacklozenge), \blacklozenge \in \{\langle sw\rangle, \langle gs\rangle, \langle ls\rangle, \langle br\rangle\}$, $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(\blacklozenge)} \mathcal{M}', w'$ implies $\mathcal{M}, w \equiv_{\mathcal{ML}(\blacklozenge)} \mathcal{M}', w'$.*

Proof. We will only prove the $\mathcal{ML}(\langle sw\rangle)$ case by structural induction.

The base case holds by (agree), and the $\wedge$ and $\neg$ cases are trivial.

[$\Diamond\varphi$ **case:**] Let $\mathcal{M} = \langle W, R, V\rangle$ and $\mathcal{M}' = \langle W', R', V'\rangle$. Suppose $\mathcal{M}, w \models \Diamond\varphi$. Then there is v in W s.t. wRv and $\mathcal{M}, v \models \varphi$. Since Z is a bisimulation, by

always	(nontriv)	Z is not empty
always	(agree)	If $(w,S)Z(w',S')$, w and w' make the same propositional variables true.
$\Diamond$	(zig)	If wSv, there is $v'\in W'$ s.t. $w'S'v'$ and $(v,S)Z(v',S')$
	(zag)	If $w'S'v'$, there is $v\in W$ s.t. wSv and $(v,S)Z(v',S')$
$\langle sw\rangle$	(sw-zig)	If wSv, there is $v'\in W'$ s.t. $w'S'v'$ and $(v,S^*_{vw})Z(v',S'^*_{v'w'})$
	(sw-zag)	If $w'S'v'$, there is $v\in W$ s.t. wSv and $(v,S^*_{vw})Z(v',S'^*_{v'w'})$
$\langle gs\rangle$	(gs-zig)	If vSu, there is $v',u'\in W'$ s.t. $v'S'u'$ and $(w,S^-_{vu})Z(w',S'^-_{v'u'})$
	(gs-zag)	If $v'S'u'$, there is $v,u\in W$ s.t. vSu and $(w,S^-_{vu})Z(w',S'^-_{v'u'})$
$\langle ls\rangle$	(ls-zig)	If wSv, there is $v'\in W'$ s.t. $w'S'v'$ and $(v,S^-_{wv})Z(v',S'^-_{w'v'})$
	(ls-zag)	If $w'S'v'$, there is $v\in W$ s.t. wSv and $(v,S^-_{wv})Z(v',S'^-_{w'v'})$
$\langle br\rangle$	(br-zig)	If $\neg wSv$, there is $v'\in W'$ s.t. $\neg w'S'v'$ and $(v,S^+_{wv})Z(v',S'^+_{w'v'})$
	(br-zag)	If $\neg w'S'v'$, there is $v\in W$ s.t. $\neg wSv$ and $(v,S^+_{wv})Z(v',S'^+_{w'v'})$

Fig. 1. Conditions for $\mathcal{ML}(\blacklozenge)$-bisimulations.

(zig) we have $v'\in W'$ s.t. $w'R'v'$ and $(v,R)Z(v',R')$. By inductive hypothesis, $\mathcal{M}',v'\models\varphi$ and by definition $\mathcal{M}',w'\models\Diamond\varphi$. For the other direction use (zag).

[$\langle sw\rangle\varphi$ **case:**] For the left to the right direction suppose $\mathcal{M},w\models\langle sw\rangle\varphi$. Then there is $v\in W$ s.t. wRv and $\mathcal{M}^*_{vw},v\models\varphi$. Because Z is a bisimulation, by (sw-zig) we have $v'\in W'$ s.t. $w'R'v'$ and $(v,R^*_{vw})Z(v',R'^*_{v'w'})$. By inductive hypothesis, $\mathcal{M}'^*_{v'w'},v'\models\varphi$ and by definition $\mathcal{M}',w'\models\langle sw\rangle\varphi$. For the other direction use (sw-zag). □

3 Expressive Power

With the appropriate notions of bisimulation at hand we can now start the comparison of the expressive power of the different dynamic modal logics we introduced. We will use the following standard definition of when a logic is at least as expressive as another.

Definition 7 ($\mathcal{L}\leq\mathcal{L}'$). *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 uncomparable *if $\mathcal{L}\not\leq\mathcal{L}'$ and $\mathcal{L}'\not\leq\mathcal{L}$.*

By inspecting suitable models we can establish the following result.

Theorem 8. *For all $\blacklozenge_1,\blacklozenge_2\in\{\langle sw\rangle,\langle gs\rangle,\langle ls\rangle,\langle br\rangle\}$ with $\blacklozenge_1\neq\blacklozenge_2$, $\mathcal{ML}(\blacklozenge_1)$ and $\mathcal{ML}(\blacklozenge_2)$ are uncomparable.*

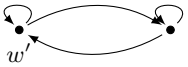
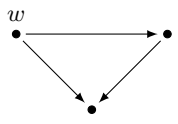
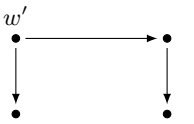
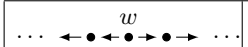
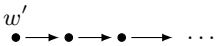
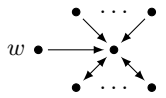
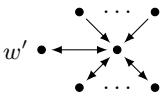
$\mathcal{M}$	$\mathcal{M}'$	Distinct by	Bisimilar for
		$\langle br\rangle\langle br\rangle\top$ $\langle gs\rangle\top$	$\mathcal{ML}(\langle ls\rangle)$ $\mathcal{ML}(\langle sw\rangle)$
		$\langle ls\rangle\Diamond\top$ $\langle gs\rangle\Diamond\top$	$\mathcal{ML}(\langle sw\rangle)$ $\mathcal{ML}(\langle br\rangle)$
		$\langle sw\rangle\langle sw\rangle\Diamond\Diamond\Diamond\Box\bot$ $[br][br]\bot$	$\mathcal{ML}(\langle gs\rangle)$ $\mathcal{ML}(\langle ls\rangle)$
		$\langle sw\rangle\Diamond\Box\bot$	$\mathcal{ML}(\langle br\rangle)$
		$\langle ls\rangle\Diamond\Box\bot$	$\mathcal{ML}(\langle gs\rangle)$

Fig. 2. Bisimilar models and distinguishing formulas.

Proof. In Figure 2 we summarize our results by presenting pairs of models that are bisimilar for a given logic and distinguishable by another. More precisely, the formulas given in the third column are false at $\mathcal{M}, w$ and true at $\mathcal{M}', w'$.

That the models are bisimilar for the given logics can be easily verified for the first two rows. In the third row, the given models are bisimilar for $\mathcal{ML}(\langle gs\rangle)$ and $\mathcal{ML}(\langle ls\rangle)$ because they are bisimilar for $\mathcal{ML}$, they are acyclic and (for $\mathcal{ML}(\langle gs\rangle)$) they contain the same number of edges. In the fourth row, both models are $\mathcal{ML}(\langle br\rangle)$-bisimilar since they are infinite, hence one can add as many links as needed to points that are modally bisimilar.

Finally, the pointed models of the last row are the same graph with a different evaluation point. The graph is a star that has infinitely many ingoing branches, and infinitely many ingoing-outgoing branches. w is a point located at the end of an ingoing branch, and w' is at the end of an ingoing-outgoing branch. Let us present the $\mathcal{ML}(\langle gs\rangle)$-bisimulation as a game between Spoiler and Duplicator. If Spoiler moves to the center of the star, Duplicator can do the same and both situations become undistinguishable. If Spoiler deletes one of the ingoing edges that has w or w' as origin, then Duplicator does the same on the other graph, and any further edge deletion can also be imitated. If Spoiler deletes the outgoing edge that goes from the center of the graph towards w', then Duplicator can delete any outgoing edge without changing the graph, given that there are infinitely many edges of both kinds. □

4 Model Checking Dynamic Logics

In this section we establish complexity results for the model checking task in the various dynamic modal logics we presented. All the results are established using a

similar argument: hardness proofs are done by encoding the satisfiability problem of Quantified Boolean Formulas (QBF) [8] as the model checking problem of each logic. While the idea behind the encoding is the same for all the logics involved, the encoding needs to be slightly modified in each case taking into consideration the semantics of the various dynamic operators.

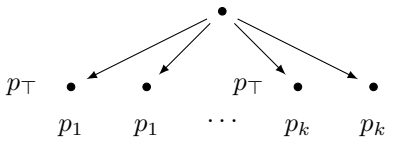
PSpace-hardness for global sabotage was already proved in [7,6], but we provide here a more direct proof.

Theorem 9. *For $\blacklozenge \in \{\langle sw\rangle, \langle gs\rangle, \langle ls\rangle, \langle br\rangle\}$, model checking for any of the logics $\mathcal{ML}(\blacklozenge)$ is PSpace-hard.*

Proof. We will reduce the PSpace-complete satisfiability problem of Quantified Boolean Formulas (QBF) to the model checking problem of each of these logics. For a complete proof of the case of $\mathcal{ML}(\langle sw\rangle)$, consult the appendix.

Consider $\mathcal{ML}(\langle sw\rangle)$. Let α be a QBF formula with variables $\{x_1, \ldots, x_k\}$. Without loss of generality we can assume that α has no free variables and no variable is quantified twice. One can build in polynomial time the relational structure $\mathcal{M}_k = \langle W, R, V\rangle$ over a signature with one relational symbol and propositions $\{p_\top, p_1, \ldots, p_k\}$, where:

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



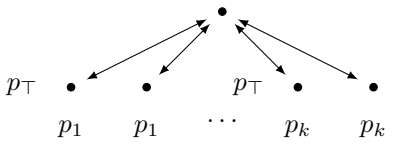
Let $(\ )'$ be the following linear translation from QBF to $\mathcal{ML}(\langle sw\rangle)$

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

It remains to see that α is satisfiable if, and only if, $\mathcal{M}_k, w \models (\alpha)'$ holds. This part of the proof is in the appendix. This shows that the model checking problem of $\mathcal{ML}(\langle sw\rangle)$ is PSpace-hard.

For $\mathcal{ML}(\langle gs\rangle)$ and $\mathcal{ML}(\langle ls\rangle)$, we use the following model:

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



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

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

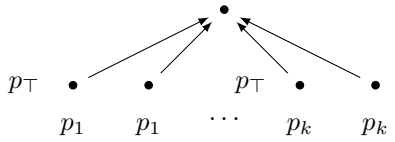
From QBF to $\mathcal{ML}(\langle gs\rangle)$, we provide the following translation:

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

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

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

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



And use the following linear translation $(\ )'$:

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

Theorem 10. *Model checking for $\mathcal{ML}(\langle sw\rangle, \langle gs\rangle, \langle ls\rangle, \langle br\rangle)$ is in PSpace.*

Proof. The evaluation of the truth of a formula in a model can be done by a polynomial space algorithm that follows Definition 3.

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

With the previous results we get:

Theorem 11. *For $\blacklozenge \in \{\langle sw\rangle, \langle gs\rangle, \langle ls\rangle, \langle br\rangle\}$, model checking for any of the logics $\mathcal{ML}(\blacklozenge)$ is PSpace-complete.*

5 Conclusions

In this article we investigate dynamic modal logics that can modify the model during the evaluation of a formula. Dynamic Epistemic Logics (DEL) as those investigated in [9,5,11,13] are well known examples of languages which can also update the model during evaluation. The standard update operation used in DELs is to move evaluation to a submodel defined by a certain 'announcement',

e.g., to the model representing the fact that φ is now known, obtained as the restriction to all the nodes satisfying a formula φ. Instead, in this article we investigate logics that can explicitly modify the accessibility relation, as the sabotage logics first introduced by van Benthem in [12].

We introduce a number of operators with both local and global effects, and which can add, delete and modify edges in the accessibility relation. The goal was to investigate the different degrees of liberty that the operators offered, and how much overlap there was between the logics they defined, and the models they could describe.

We show in Sections 2 and 3 that the languages obtained by the extension of the basic modal logic with each of the dynamic operators can be characterized using bisimulations. Actually, even though each operator requires a particular pair of zig and zag conditions, the definition is modular and the set up homogeneous. All the bisimulations involved are of the same type, linking a pair of point of evaluation and accessibility relation in one model, with a similar pair in the other. Moreover, a suitable definition of bisimulation for the basic modal logic extended with any combination of the new dynamic operators can be obtained by using the adequate zig and zag conditions associated to the operators involved. Summing up then, even though the logics obtained are different in each case, they are all amenable to fairly classical modal analysis.

In Section 4 we turn to model checking, and show that the complexity of this reasoning task is PSpace-complete for all the logics considered. Once more, the proofs are fairly homogeneous in all cases. The general set up is the encoding of the PSpace-complete QBF satisfiability problem in each of the logics. In each case, a suitable representation for the assignment and the concrete translation used needs to be defined, but once this is done the proof is similar.

More precisely, we established the complexity of the *combined* model checking task, measured in function of the length of an input model and an input formula. It is also possible to consider the task of model checking against a fixed model, measuring its complexity in function of the size of an input formula (this is known as the formula complexity). One can also fix a formula and measure the complexity of model checking in function of the length of an input model (known as the program complexity or data complexity). Both notions were introduced in [14], and it has been shown in [6] that the formula complexity and the program complexity of $\mathcal{ML}(\langle gs \rangle)$ are respectively linear and polynomial. We believe that the proof generalizes to $\mathcal{ML}(\langle ls \rangle)$, $\mathcal{ML}(\langle sw \rangle)$ and $\mathcal{ML}(\langle br \rangle)$ with identical results.

Another natural direction for future research would be to investigate the complexity of the satisfiability problem of these logics. From [6], we already know that $\mathcal{ML}(\langle gs \rangle)$ is undecidable. We conjecture that using techniques from [3,1], it is possible to prove that the problem is undecidable in all remaining cases.

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".

References

1. Areces, C., Figueira, D., Figueira, S., Mera, S.: The expressive power of memory logics. Review of Symbolic Logic 4(2), 290–318 (2011)
2. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press (2001)
3. Blackburn, P., Seligman, J.: Hybrid languages. Journal of Logic, Language and Information 4, 251–272 (1995)
4. Blackburn, P., Wolter, F., van Benthem, J. (eds.): Handbook of Modal Logics. Elsevier (2006)
5. Gerbrandy, J.: Bisimulations on Planet Kripke. Ph.D. thesis, University of Amsterdam (1999), ILLC Dissertation series DS-1999-01
6. Löding, C., Rohde, P.: Model checking and satisfiability for sabotage modal logic. In: Pandya, P., Radhakrishnan, J. (eds.) FSTTCS. Lecture Notes in Computer Science, vol. 2914, pp. 302–313. Springer (2003)
7. Löding, C., Rohde, P.: Solving the sabotage game is PSPACE-hard. In: Mathematical Foundations of Computer Science 2003, Lecture Notes in Computer Science, vol. 2747, pp. 531–540. Springer, Berlin (2003)
8. Papadimitriou, C.: Computational Complexity. Addison-Wesley (1994)
9. Plaza, J.: Logics of public communications. Synthese 158(2), 165–179 (2007)
10. Rohde, P.: On games and logics over dynamically changing structures. Ph.D. thesis, RWTH Aachen (2006)
11. van Benthem, J.: Logics for information update. In: TARK'01: Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge. pp. 51–67. Morgan Kaufmann Publishers Inc. (2001)
12. van Benthem, J.: An essay on sabotage and obstruction. In: Mechanizing Mathematical Reasoning. pp. 268–276 (2005)
13. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Kluwer (2007)
14. Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: Lewis, H.R., Simons, B.B., Burkhard, W.A., Landweber, L.H. (eds.) STOC. pp. 137–146. ACM (1982)

Appendix

Theorem 4. *$\mathcal{ML}(\blacklozenge)$ does not have the tree model property, for $\blacklozenge \in \{\langle sw\rangle, \langle gs\rangle, \langle ls\rangle, \langle br\rangle\}$*

Proof. We are going to list formulas that force no-treelike models:

1. The formula
$$\varphi = p \wedge (\bigwedge_{1\leq i\leq 3} \Box^i \neg p) \wedge \langle sw\rangle \Diamond \Diamond p$$
is true at a state w in a model, only if w has a reflexive successor.

Suppose we evaluate φ at some state w of an arbitrary model. The 'static' part of the formula $p \wedge (\bigwedge_{1 \leq i \leq 3} \Box^i \neg p)$ makes sure that p is true in w and that no p state is reachable within three steps from w (in particular, w cannot be reflexive).
Because $\langle sw \rangle \Diamond \Diamond p$ is true at w, there should be an R-successor v where $\Diamond \Diamond p$ holds once the accessibility relation has been updated to R^*_{vw}. That is, v has to reach a p-state in exactly two R^*_{vw}-steps. But the only p-state sufficiently close is w which is reachable in one step. As w is not reflexive, v has to be reflexive so that we can linger at v for one loop and reach p in the correct number of states.

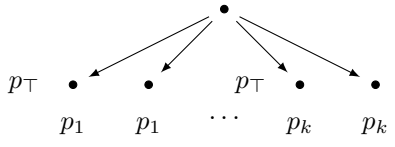
2. The formula
$$\varphi = \Diamond \Diamond \top \ \wedge \ [ls] \Box \bot$$
is true at a state w in a model, only if w is reflexive.
Suppose we evaluate φ at some state w of an arbitrary model. On one hand, the 'static' part of the formula $\Diamond \Diamond \top$ ensures it is possible to take two accessibility relations. On the other hand, the 'dynamic' part of the formula $[ls] \Box \bot$ tells us that after taking any accessibility relation and eliminating it, it is no longer possible to go anywhere else. This can only happen if the point w is reflexive and does not have any other outgoing links.
3. The formula
$$\varphi = \Diamond \Diamond \top \ \wedge \ [gs] \Box \bot$$
(from [6]) is true at a state w in a model, only if w is reflexive.
4. The formula
$$\varphi = \langle br \rangle \Box \bot$$
is only satisfiable in models that have at least two unconnected points. □

Theorem 11. *Model checking for $\mathcal{ML}(\langle sw \rangle)$ is PSpace-hard.*

Proof. We will reduce the PSpace-complete satisfiability problem of Quantified Boolean Formulas (QBF) to the model checking problem of $\mathcal{ML}(\langle sw \rangle)$.

Let α be a QBF formula with variables $\{x_1, \ldots, x_k\}$. Without loss of generality we can assume that α has no free variables and no variable is quantified twice. One can build in polynomial time the relational structure $\mathcal{M}_k = \langle W, R, V \rangle$ over a signature with one relational symbol and propositions $\{p_\top, p_1, \ldots, p_k\}$, where:

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



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

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

It remains to see that α is satisfiable iff $\mathcal{M}_k, w \models (\alpha)'$ holds. Let us write $v \models_{\text{qbf}} \alpha$ if valuation $v : \{x_1, \ldots, x_k\} \to \{0, 1\}$ satisfies α. For a model $\mathcal{M}$ with relation R we define $v_R : \{x_1, \ldots, x_k\}$ as "$v_R(x_i) = 1$ iff $(w, w_i^1) \notin R$", in the present case, iff the link between w and w_i^1 has been swapped.

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

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

The boolean cases follow directly from the inductive hypothesis.

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