Swap Logic

Carlos Areces[a,b], Raul Fervari[a], and Guillaume Hoffmann[a]

[a] FaMAF, Universidad Nacional de Córdoba, Argentina
[b] CONICET, Argentina
{areces, fervari, hoffmann}@famaf.unc.edu.ar

Abstract. We investigate dynamic modal operators that can change the model during evaluation. We define the logic $\mathcal{SL}$ by extending the basic modal language with the $\overleftrightarrow{\Diamond}$ modality, which is a diamond operator that in addition has the ability to invert pairs of related elements in the domain while traversing an edge of the accessibility relation.
$\mathcal{SL}$ is very expressive: it fails to have the finite and the tree model property. We show that $\mathcal{SL}$ is equivalent to a fragment of first-order logic by providing a satisfiability preserving translation. In addition, we provide an equivalence preserving translation from $\mathcal{SL}$ to the hybrid logic $\mathcal{H}(:,\downarrow)$. We also define a suitable notion of bisimulation for $\mathcal{SL}$ and investigate its expressive power, showing that it lies strictly between the basic modal logic and $\mathcal{H}(:,\downarrow)$. We finally show that its model checking problem is PSpace-complete and its satisfiability problem is undecidable.

Keywords: Modal logic, dynamic logics, expressivity, complexity.

1 Changing the Model

Modal logics [6,8] 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 $\mathcal{BML}$ 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 *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 the 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 [19].

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

$$\mathcal{M}, w \models \Diamond\varphi \quad \text{iff} \quad \text{there is a state } v \neq w \text{ of } \mathcal{M} \text{ such that } \mathcal{M} \setminus \{v\}, w \models \varphi.$$

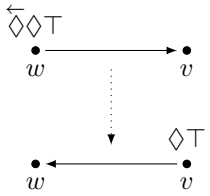
As a modal logic, it is clear that the $\Diamond$ operator *changes* the model in which a formula is evaluated. As van Benthem puts it, $\Diamond$ is an "external" modality that takes evaluation to another model, obtained from the current one by deleting some state or transition. Various sabotage modal logics have been studied [13,12,17]. In particular, it has been proved that solving the sabotage game is PSpace-hard, while the model checking problem of the associated modal logic is PSpace-complete and the satisfiability problem is undecidable. The logic fails to have both the finite model property and the tree model property. A translation of the sabotage modal logic into first-order logic is also provided.

Memory logics, investigated in [1,14,3,4], are another family of modal logics that can change models. The semantics of these languages 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. The general properties of memory logics are similar to those of sabotage logics: a PSpace-complete model checking problem, an undecidable satisfiability problem, and failure of both the finite model and the tree model properties.

In this article, we will investigate a modal changing operator that neither shrinks nor expands the model[1]. Instead, it has the ability to swap the direction of a traversed edge. The $\overleftarrow{\Diamond}$ 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 $\overleftarrow{\Diamond}$. The formula $\overleftarrow{\Diamond}\Diamond\top$ is true in a model with two related states:

[1] We will see in the next section that this is not completely true, the accessibility relation might shrink but only in particular cases.



As we can see in the picture, evaluation starts at state w with the edge pointing from w to v, but after evaluating the $\overleftarrow{\Diamond}$ operator, it continues at state v with the edge now pointing from v to w. In this article, we will see that the swap logic $\mathcal{SL}$ obtained by adding $\overleftarrow{\Diamond}$ to the basic modal logic fares as the other model changing logics we mentioned above: thought still a fragment of first-order logic, it is very expressive with a PSpace-complete model checking problem and an undecidable satisfiability problem.

The rest of the article is organized as follows. We first present syntax and semantics of $\mathcal{SL}$ in Section 2. In Section 3 we define a satisfiability preserving translation of $\mathcal{SL}$ into first-order logic, then in Section 4 we provide an equivalence preserving translation into hybrid logic. In Section 5 we study the expressive power of $\mathcal{SL}$ using bisimulations. Finally, in Section 6 we study the complexity of its model checking and satisfiability problems. We close the article in Section 7 summing up our results.

2 Basic Definitions

Now we introduce syntax and semantics for $\mathcal{SL}$. We define some notation that will help us describe models with swapped accessibility relations. We provide some examples of $\mathcal{SL}$ formulas, and we close this section showing the lack of finite model property of the logic.

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

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

where $p \in \mathsf{PROP}$ *and* $\varphi, \psi \in \mathsf{FORM}$. *Other operators are defined as usual. In particular* $\Box\varphi$ *is a shorthand for* $\neg\Diamond\neg\varphi$ *and* $\overleftarrow{\Box}\varphi$ *stands for* $\neg\overleftarrow{\Diamond}\neg\varphi$.

The above definition shows that $\mathcal{SL}$ is the syntactic extension of the basic modal logic with the $\overleftarrow{\Diamond}$ operator. This is also true semantically: formulas of $\mathcal{SL}$ are evaluated in standard relational models, and the meaning of all the operators of the basic modal logic is unchanged. When we evaluate formulas of $\mathcal{SL}$ containing the $\overleftarrow{\Diamond}$ operator, we will need to keep track of the edges that have been swapped. 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 Variants). *A model $\mathcal{M}$ is a triple $\mathcal{M} = \langle W, R, V\rangle$, where W is a non-empty set whose elements are called points or states; $R \subseteq W \times W$ is the accessibility relation; and $V : \mathsf{PROP} \mapsto \mathcal{P}(W)$ is a valuation.*

Given a model $\mathcal{M} = \langle W, R, V\rangle$, we define the model $\mathcal{M}^*_{vw} = \langle W, R^*_{vw}, V\rangle$, where $R^*_{vw} = (R \setminus wv) \cup vw$, $wv \in R$.

We will refer to models obtained by swapping edges, or more generally, modifying a given initial model, as model variants*.*

Definition 3 (Semantics). *Let w be a state in $\mathcal{M}$, the pair $(\mathcal{M}, w)$ is a pointed model; we usually drop parenthesis and call $\mathcal{M}, w$ a pointed model. Given a pointed model $\mathcal{M}, w$ and a formula φ we say that $\mathcal{M}, w$ satisfies φ (notation, $\mathcal{M}, w \models \varphi$) when*

$$
\begin{array}{ll}
\mathcal{M}, w \models p & \textit{iff } w \in V(p) \\
\mathcal{M}, w \models \neg\varphi & \textit{iff } \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \textit{iff } \mathcal{M}, w \models \varphi \textit{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \Diamond\varphi & \textit{iff for some } v \in W \textit{ s.t. } (w, v) \in R, \mathcal{M}, v \models \varphi \\
\mathcal{M}, w \models \overleftarrow{\Diamond}\varphi & \textit{iff for some } v \in W \textit{ s.t. } (w, v) \in R, \mathcal{M}^*_{vw}, v \models \varphi.
\end{array}
$$

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

The semantic condition for $\overleftarrow{\Diamond}$ looks quite innocent but, as we will see in the next couple of examples, it is actually very expressive. First, let us see the operator in action in two simple important cases.

Example 1. The $\overleftarrow{\Diamond}$ operator leaves reflexive edges unchanged:

Example 2. The $\overleftarrow{\Diamond}$ operator collapses symmetric edges into a single one:

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

Now let us see a couple of examples where $\mathcal{SL}$ formulas can force model with complex structures.

Example 3. Define $\Box^0\varphi$ as φ, $\Box^{n+1}\varphi$ as $\Box\Box^n\varphi$, and let $\Box^{(n)}\varphi$ be a shorthand for $\bigwedge_{1 \leq i \leq n} \Box^i\varphi$. The formula

$$\varphi = p \wedge \Box^{(3)}\neg p \wedge \overleftarrow{\Diamond}\Diamond\Diamond p$$

is true at a state w in a model if and only if w has a reflexive successor. Notice that no equivalent formula exists in the basic modal language (formulas in the basic modal language can always be satisfied at the root of a tree model).

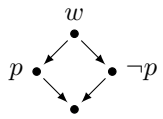
Let us analyse the formula in detail. Suppose we evaluate φ at some state w of an arbitrary model. The 'static' part of the formula $p \wedge \Box^{(3)}\neg p$ makes sure that p is true in w and that no p state is reachable within a three step neighbourhood of w (in particular, w cannot be reflexive). Now, the 'dynamic' part of the formula $\overleftarrow{\Diamond}\Diamond\Diamond p$ will do its magic. Because $\overleftarrow{\Diamond}\Diamond\Diamond p$ 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}. Now, v has to reach a p-state in exactly two R^*_{vw}-steps to satisfy $\Diamond\Diamond p$. But the only p state sufficiently close for this to happen 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.

We analyse now two classical properties of modal logics: the tree and the finite model property. Both fail in the case of $\mathcal{SL}$.

Theorem 1. *The tree model property fails for $\mathcal{SL}$, i.e., there are formulas in $\mathcal{SL}$ that cannot be satisfied at the root of a tree.*

Proof. Consider the following $\mathcal{SL}$ formula that forces models that contain a diamond:

$$\begin{aligned}\varphi = {} & \Diamond p \wedge \Diamond\neg p \wedge \Box\Diamond\top \wedge \Box\Box\Box\bot \\ & \wedge \overleftarrow{\Box}\overleftarrow{\Box}\Box\Box\bot \\ & \wedge \overleftarrow{\Box}\Box\Box\bot \\ & \wedge \overleftarrow{\Diamond}\overleftarrow{\Diamond}\Diamond\Diamond\Diamond\Diamond\top\end{aligned}$$



When evaluated at w, the first line of φ forces two different successors w_1 and w_2; each of them has successors which are dead ends. The second line forces w to have at most two successors, while the third makes sure that w_1 and w_2 have at most one successor. Finally, the last line says that starting from w and after swapping two successive edges, we can create a path of length at least five. For that to be true, w_1 and w_2 must share a successor. □

Theorem 2. *The finite model property fails for $\mathcal{SL}$, i.e., there are formulas in $\mathcal{SL}$ that can only be satisfied in infinite models.*

Proof. Consider the following $\mathcal{SL}$ formula:

$$\begin{aligned}\varphi = {} & s \wedge \Box^{(9)}\neg s && (1)\\ & \wedge\Diamond\top && (2)\\ & \wedge\Box\Diamond\top && (3)\\ & \wedge\overleftarrow{\Box}\Box\neg\Diamond s && (4)\\ & \wedge\overleftarrow{\Box}\overleftarrow{\Box}(\neg s \rightarrow \Diamond\Diamond\Diamond\Diamond s) && (5)\\ & \wedge\overleftarrow{\Box}\overleftarrow{\Box}\overleftarrow{\Box}(\neg\Diamond s \rightarrow \Diamond\Diamond\Diamond(\neg s \wedge \Diamond\Diamond\Diamond s)) && (6)\end{aligned}$$

The formula forces models to have an infinite chain of states. It does so by ensuring that the propositional symbol s is true only in the evaluation state w, and false in all the states reachable from w. Then, it enforces specific properties on the model, locating states by their distance to w using formulas of the form $\Diamond \dots \Diamond s$ after swapping an outgoing edge from w. This way it is possible to enforce seriality, irreflexivity and transitivity on a chain of states. Let us check the formula in detail.

(1) makes s true at the state w and false at all states accessible within 9 steps. By (2), w has one successor and by (3), every successor of w has some successor. (4) makes every successor of w irreflexive.

(5) tells that from any state $v \neq w$ reachable in two swapping steps, it is possible to go back to w in five steps. But this is only possible by first going to w in two steps, then going to v in one step and going again to w in two steps. Hence all states accessible in two steps from w are also accessible in one. This makes w a 'spy state', i.e., it is directly connected to every state in the submodel generated from it. (6) enforces the same property on the successors of w.

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

These examples should warn us about the expressivity of $\mathcal{SL}$, which is certainly well above that of the basic modal language. We might even wonder if the dynamic characteristics of $\mathcal{SL}$ can be captured inside first-order logic.

3 Swap Logic as a Fragment of First-Order Logic

As we saw in the previous section $\mathcal{SL}$ is very expressive. This expressivity is intrinsically tied to its ability to modify the model during evaluation (remember that we only extended the basic modal language with a harmless looking diamond modality). It is well known that the basic modal language can be seen as a fragment of first-order logic via a satisfiability preserving translation. We might wonder if this is the case also for $\mathcal{SL}$. It is not obvious, a priori, that the model changing capabilities of $\mathcal{SL}$ can be captured in the static setting of first-order.

In this section we provide a satisfaction preserving translation of $\mathcal{SL}$ into $\mathcal{FOL}$, but it is far more involved that the standard translation used for the basic modal logic. Indeed, we will have to force the translated formula to be evaluated in a particular class of models. We will do this through the use of two sorted first-order logic [22]. First, let us introduce formally the syntax and semantics of (sorted) first-order logic.

Definition 4 (Sorted first-order language). *Let* $\mathsf{REL} = \{R_1, R_2, \dots\}$ *be a countable set of relation symbols,* $\mathsf{FUN} = \{f_1, f_2, \dots\}$ *a countable set of function symbols,* $\mathsf{CON} = \{c_1, c_2, \dots\}$ *a countable set of constant symbols,* $\mathsf{VAR} = \{x_1, x_2, \dots\}$ *a countable set of variables and* $\mathsf{SORT} = \{O_1, , O_2, \dots\}$ *a countable set of sorts. We assume that* REL, FUN, CON, VAR *and* SORT *are pairwise disjoint. To each relation symbol* $R_i \in \mathsf{REL}$ *and each function symbol* $f_i \in \mathsf{FUN}$ *we associate an arity* $n > 0$ *and a sort type given by the function* S*, where*

$S(R_i) \in \mathsf{SORT}^n$ *and* $S(f_i) \in (\mathsf{SORT}^n, \mathsf{SORT})$. *To each variable* $x \in \mathsf{VAR}$ *and constant* $c \in \mathsf{CONS}$, *we have that* $S(x) \in \mathsf{SORT}$ *and* $S(c) \in \mathsf{SORT}$. *We call* $\mathcal{S} = \langle \mathsf{REL}, \mathsf{FUN}, \mathsf{CON}, \mathsf{VAR}, \mathsf{SORT} \rangle$ *a signature. If* $\mathsf{FUN} = \{\}$ *we will say that the signature is relational.*

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

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

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

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

where $t_1, t_2, \ldots, t_n \in \mathsf{TERM}$, $R_i \in \mathsf{REL}$ *is an* n*-ary relation symbol,* $\varphi, \varphi_1, \varphi_2 \in \mathsf{FORM}$, $x_i \in \mathsf{VAR}$, $S(t_1) = S(t_2)$, $(S(t_1) \ldots S(t_n)) = S(R_i)$ *and* $O = S(x)$.

As usual, we take $\vee, \rightarrow, \leftrightarrow$ *and* $\forall$ *as defined symbols.*

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

Definition 5 (Sorted first-order models and satisfiability). *Let* $\mathcal{S}$ *be a signature, a sorted first-order model for* $\mathcal{S}$ *is a structure* $\mathcal{M} = \langle M, \mathcal{I} \rangle$ *with* $M = \{M_1, \ldots, M_k\}$, *where* $M_i's$ *are non-empty sets and* $\mathcal{I}$ *is an interpretation function defined over* $\mathsf{REL} \cup \mathsf{FUN} \cup \mathsf{CONS} \cup \mathsf{SORT}$ *such that:*

- *To each* $O_i \in \mathsf{SORT}$, $\mathcal{I}(O_i) \in M$.
- *To each* $R_i \in \mathsf{REL}$ *such that* $S(R_i) = (O_1, \ldots, O_n)$, $\mathcal{I}$ *assigns a relation* $\mathcal{I}(R_i) \subseteq \mathcal{I}(O_i) \times \ldots \times \mathcal{I}(O_n)$.
- *To each* $f_i \in \mathsf{FUN}$ *such that* $S(f_i) = (O_1, \ldots, O_n, O_{n+1})$, $\mathcal{I}$ *assigns a function* $\mathcal{I}(f_i) : \mathcal{I}(O_1) \times \ldots \times \mathcal{I}(O_n) \rightarrow \mathcal{I}(O_{n+1})$.
- *To each* $c_i \in \mathsf{CONS}$ *with* $S(c_i) = O_i$, $\mathcal{I}$ *assigns some element* $\mathcal{I}(c_i) \in \mathcal{I}(O_i)$.

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

$$\mathcal{I}(x_i) = g(x_i)$$
$$\mathcal{I}(f(t_1, \ldots, t_n)) = \mathcal{I}(f)(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)).$$

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

$$\begin{aligned}
\mathcal{M} &\models \top[g] && \textit{always} \\
\mathcal{M} &\models t_1 = t_2[g] && \textit{iff } \mathcal{I}(t_1) = \mathcal{I}(t_2) \\
\mathcal{M} &\models R(t_1, \ldots, t_n)[g] && \textit{iff } (\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)) \in \mathcal{I}(R) \\
\mathcal{M} &\models \neg\varphi[g] && \textit{iff } \mathcal{M} \not\models \varphi[g] \\
\mathcal{M} &\models \varphi_1 \wedge \varphi_2[g] && \textit{iff } \mathcal{M} \models \varphi_1[g] \textit{ and } \mathcal{M} \models \varphi_2[g] \\
\mathcal{M} &\models \exists_{x:O}.\varphi[g] && \textit{iff } \mathcal{M} \models \varphi[g^x_m] \textit{ for some } m \in \mathcal{I}(O).
\end{aligned}$$

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

In the rest of the paper we will call $2\mathcal{FOL}$, to the sorted first-order logic with two sorts.

The key idea to provide a translation from $\mathcal{SL}$ to $2\mathcal{FOL}$ is to unravel dynamic $\mathcal{SL}$ models so that they can be represented by static first-order models. Consider the following $\mathcal{SL}$ model

Swap operators will be able to transform the model in specific ways. A $\overleftarrow{\Diamond}$ operator evaluated at w will produce a model variant where the accessibility relation is just the (v, w) edge, and similarly for a $\overleftarrow{\Diamond}$ operator evaluated at v.

As illustrated in Figure 1, we can use an index to identify each of these possible model variants, and then connect them using a new accesibility relation, where some state w in the model i_j is related with some state v in i_k, if $(w, v) \in R$ in the model i_j, and i_k is identical to i_j except that the edge (w, v) has been swapped around. The R relation is meant to link states with the same index, while the S relation is meant to link states from a given model variant to a possibly different model variant corresponding to the result of swapping an edge.

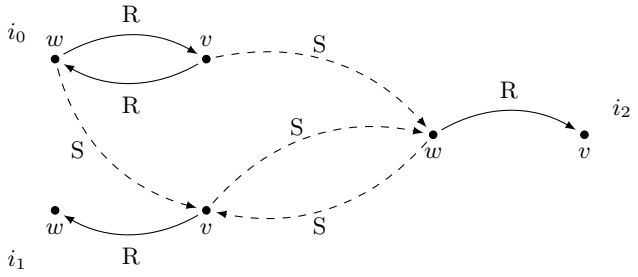


Fig. 1. A $2\mathcal{FOL}$ model that statically represents a dynamic $\mathcal{SL}$ model.

Now, we apply this idea to translate formulas of $\mathcal{SL}$ into $2\mathcal{FOL}$. We start by taking the well-known standard translation from $\mathcal{BML}$ to $\mathcal{FOL}$, but extend its definition so that $\overleftarrow{\Diamond}$ is translated and interpreted using the S relation. To ensure that the relations R and S behave as intended, we add to the translation some additional conditions. Let us see the translation in detail.

Definition 6 (Standard Translation). *We define $ST : \mathcal{SL} \times \mathsf{VAR} \times \mathsf{IND} \to 2\mathcal{FOL}$ where VAR is a set of variables and IND a set of indexes. It produces*

formulas in two sorted first-order logic with sorts W and I, one 3-ary predicate symbol R' over $W \times I \times W$, one 4-ary predicate symbol S' over $W \times I \times W \times I$, and one unary predicate symbol P_p over W for each $p \in \mathsf{PROP}$.

$$\begin{array}{ll} ST(p,x,i) & = P_p(x) \\ ST(\neg\varphi,x,i) & = \neg ST(\varphi,x,i) \\ ST(\varphi\wedge\psi,x,i) & = ST(\varphi,x,i) \wedge ST(\psi,x,i) \\ ST(\Diamond\varphi,x,i) & = \exists_{y:W}.(R'(x,i,y) \wedge ST(\varphi,y,i)) \\ ST(\overleftarrow{\Diamond}\varphi,x,i) & = \exists_{y:W,j:I}.(S'(x,i,y,j) \wedge ST(\varphi,y,j)). \end{array}$$

Define the following two $2\mathcal{FOL}$ formulas:

$$\begin{array}{l} Cond_1 = \forall_{w,v:W,i:I}.(R'(w,i,v) \leftrightarrow \exists_{j:I}.S'(w,i,v,j)) \\ Cond_2 = \forall_{w,v:W,i,j:I}. \\ \qquad (S'(w,i,v,j) \ \leftrightarrow \\ \qquad\quad (R'(w,i,v) \\ \qquad \wedge\ R'(v,j,w) \\ \qquad \wedge\ w \neq v \ \rightarrow\ \neg R'(w,j,v) \\ \qquad \wedge\ \forall_{w',v':W}.(w,v)\neq(w',v') \wedge (w,v)\neq(v',w') \\ \qquad\qquad\qquad \rightarrow\ R'(w',i,v') \leftrightarrow R'(w',j,v')) \ \ ). \end{array}$$

We finally define $ST'(\varphi,x,i) = ST(\varphi,x,i) \wedge Cond_1 \wedge Cond_2$.

$Cond_1$ says that whenever two states are linked in some model variant then it is possible to swap the edge between them. That is, there exists some model variant indexed by j such that the state w in model variant i is linked to the state v in model variant j. The other direction says that if two points are linked by S between possibly different model variants, then they should be linked by R in the same model variant.

$Cond_2$ enforces a set of conditions that correspond to the expected effect of swapping edges. Let us read it from left to right. Assume a 'swapping edge' exists between state w of model variant i and state v of model variant j. Then there must be an edge in the model variant i between w and v ($R'(w,i,v)$). There is also an edge from v to w in the model variant j ($R'(v,j,w)$). In the case where w and v are distinct states, there is no longer an edge from w to v in the model variant j. Finally, all the other edges of model variants i and j should not be affected by the swapping (this is the last two lines of $Cond_2$). The other direction of $Cond_2$ tells that whenever all these conditions are met, there should be a corresponding edge in S.

Notice that the translation from $\mathcal{SL}$ to $2\mathcal{FOL}$ is polynomial. However, the class of models in which we evaluate formulas that are the result of this translation is not the same in which we evaluate the original $\mathcal{SL}$-formula. The following definition associates each $\mathcal{SL}$ model with its corresponding $2\mathcal{FOL}$ model.

Definition 7 (Model Translation). *Let $(\ \ )'$ be the function that for every model $\mathcal{M} = \langle W,R,V\rangle$ associates a two sorted first-order model $(\mathcal{M})' =$*

$\langle \{W', I'\}, \mathcal{I}' \rangle$ *where* W', I' *and* $\mathcal{I}'$ *are defined as follows:*

$$\begin{array}{ll} W' & = W \\ I' & = \{R\} \cup \{U^*_{vw} \mid U \in I' \wedge (w,v) \in U\} \\ \mathcal{I}'(P_p) & = V(p) \\ \mathcal{I}'(R') & = \{(w,T,v) \mid T \in I', (w,v) \in T\} \\ \mathcal{I}'(S') & = \{(w,T,v,T^*_{vw}) \mid T \in I', (w,v) \in T\}. \end{array}$$

The model translation above exhaustively expands an $\mathcal{SL}$ model into a complete, static $2\mathcal{FOL}$ model that contains every possible model variant obtained by swapping arbitrary edges of the initial model. This leads to the first part of the equisatisfiability theorem. We want to show that if an $\mathcal{SL}$ formula is satisfiable, then its translation is also satisfiable.

Theorem 3. *Let* φ *be an* $\mathcal{SL}$ *formula. If* φ *is satisfiable, then* $ST'(\varphi, x, i)$ *is satisfiable.*

Proof. We will prove the following. Let $\mathcal{M} = \langle W, R, V \rangle$ be a model and φ a formula of $\mathcal{SL}$. Then

$$\mathcal{M}, w \models \varphi \text{ iff } (\mathcal{M})' \models ST'(\varphi, x, i)[x := w, i := \emptyset]. \qquad \otimes$$

In the rest of the paper we use the following notation: $\boldsymbol{j} = \mathcal{I}'(j)$, $\boldsymbol{i} = \mathcal{I}'(i)$, $\boldsymbol{w} = \mathcal{I}'(w)$, $\boldsymbol{v} = \mathcal{I}'(v)$, $\boldsymbol{w'} = \mathcal{I}'(w')$, $\boldsymbol{v'} = \mathcal{I}'(v')$.

(1) We first prove that $(\mathcal{M})' \models Cond_1 \wedge Cond_2$.

1. $\forall_{w,v:W,i:I}.(R'(w,i,v) \leftrightarrow \exists_{j:I}.S'(w,i,v,j))$
   The right-to-left direction holds from the definitions of $\mathcal{I}'(S')$ and $\mathcal{I}'(R')$. For the left-to-right direction, it suffices to choose $\boldsymbol{j} = \boldsymbol{i}^*_{\boldsymbol{vw}}$.
2. $\forall_{w,v:W,i,j:I}.(S'(w,i,v,j) \;\leftrightarrow$
   $(R'(w,i,v) \;\wedge\; R'(v,j,w) \;\wedge\; w \neq v \rightarrow \neg R'(w,j,v)$
   $\wedge\; \forall_{w',v':W}.(w,v) \neq (w',v') \wedge (w,v) \neq (v',w') \rightarrow$
   $R'(w',i,v') \leftrightarrow R'(w',j,v')))$

   Let $\mathcal{M}$ be a $\mathcal{SL}$ model such that $(\mathcal{M})' \models S'(w,i,v,j)[w := \boldsymbol{w}, i := \boldsymbol{i}, v := \boldsymbol{v}, j := \boldsymbol{j}]$ for arbitrary $\boldsymbol{w}, \boldsymbol{v}$ in W and $\boldsymbol{i}, \boldsymbol{j}$ in I. By definition of $\mathcal{I}'(S')$, we have that $(\boldsymbol{w}, \boldsymbol{v}) \in \boldsymbol{i}$ and $\boldsymbol{j} = \boldsymbol{i}^*_{\boldsymbol{vw}}$.
   - We have $(\boldsymbol{w}, \boldsymbol{v}) \in \boldsymbol{i}$ (definition of $\mathcal{I}'(R')$), hence $(\mathcal{M})' \models R'(w,i,v)$.
   - We have $(\boldsymbol{v}, \boldsymbol{w}) \in \boldsymbol{i}^*_{\boldsymbol{vw}}$, that is $(\boldsymbol{v}, \boldsymbol{w}) \in \boldsymbol{j}$, i.e., $(\mathcal{M})' \models R'(v,j,w)$.
   - Suppose $(\mathcal{M})' \models w \neq v$, that is, $\boldsymbol{w} \neq \boldsymbol{v}$. Then by definition of the $*$ operation, $(\boldsymbol{w}, \boldsymbol{v}) \notin \boldsymbol{i}^*_{\boldsymbol{vw}}$, i.e., $(\boldsymbol{w}, \boldsymbol{v}) \notin \boldsymbol{j}$, i.e., $(\mathcal{M})' \models \neg R'(w,j,v)$.
   - We have $(\mathcal{M})' \models (w,v) \neq (w',v') \wedge (w,v) \neq (v',w')$ for some $w', v' \in W$ if, and only if $(\boldsymbol{w}, \boldsymbol{v}) \neq (\boldsymbol{w'}, \boldsymbol{v'})$ and $(\boldsymbol{w}, \boldsymbol{v}) \neq (\boldsymbol{v'}, \boldsymbol{w'})$, which gives $(\boldsymbol{w'}, \boldsymbol{v'}) \in \boldsymbol{i}$ if, and only if, $(\boldsymbol{w'}, \boldsymbol{v'}) \in \boldsymbol{i}^*_{\boldsymbol{vw}}$ (by definition of the $*$ operation). This is equivalent to $(\boldsymbol{w'}, \boldsymbol{v'}) \in \boldsymbol{j}$, therefore $(\mathcal{M})' \models R'(w',i,v') \leftrightarrow R'(w',j,v')$.

Hence $(\mathcal{M})' \models Cond_1 \wedge Cond_2$.

(2) Now let us prove the following property by structural induction:

$$\langle W, S, V\rangle, w \models \varphi \text{ iff } (\mathcal{M})' \models ST(\varphi, x, i)[x := w, i := S].$$

When examining the following inductive cases, we will prove the left-to-right implication only. The right-to-left implication can be proved following the steps in the converse direction.

$\varphi = p$: trivial by definition of $(\ )'$.
$\varphi = \neg\psi$ and $\varphi = \psi_1 \wedge \psi_2$: trivial.
$\varphi = \Diamond\psi$: Suppose $\langle W, S, V\rangle, w \models \Diamond\psi$. We have:

- $\exists v \in W$ s.t. $S(w, v)$ and $\langle W, S, V\rangle, v \models \psi$, by definition of $\models \quad \otimes_1$
- $(w, S, v) \in \mathcal{I}'(R')$, from $\otimes_1$ by definition of $\mathcal{I}'(R')$
- $\langle\{W', I'\}, \mathcal{I}'\rangle \models R'(x, i, y)[x := w, y := v, i := S]$, by definition of $(\ )'$
- $\langle\{W', I'\}, \mathcal{I}'\rangle \models ST(\psi, y, i)[y := v, i := S]$, from $\otimes_1$ by I.H.
- $\langle\{W', I'\}, \mathcal{I}'\rangle \models \exists_{y:W}.R'(x, i, y) \wedge ST(\psi, y, i)[x := w, i := S]$,
- $(\mathcal{M})' \models ST(\Diamond\psi, x, i)[x := w, i := S]$, by definition of ST.

$\varphi = \overleftarrow{\Diamond}\psi$: Suppose $\langle W, S, V\rangle, w \models \overleftarrow{\Diamond}\psi$. We have:

- $\exists v \in W$ s.t. $S(w, v)$ and $\langle W, S^*_{vw}, V\rangle, v \models \psi$, by definition of $\models \otimes_2$
- $(w, S, v, S^*_{vw}) \in \mathcal{I}'(S')$, by $\otimes_2$ and definition of $\mathcal{I}'$
- $\langle\{W', I'\}, \mathcal{I}'\rangle \models S'(x, i, y, j)[x := w, i := S, y := v, j := S^*_{vw}]$
- $\langle\{W', I'\}, \mathcal{I}'\rangle \models ST(\psi, y, j)[y := v, j := S^*_{vw}]$, from $\otimes_2$ by I.H.
- $\langle\{W', I'\}, \mathcal{I}'\rangle \models S'(x, i, y, j) \wedge ST(\psi, y, j)[x := w, i := S, y := v, j := S^*_{vw}]$
- $\langle\{W', I'\}, \mathcal{I}'\rangle \models \exists_{y:W,j:I}.S'(x, i, y, j) \wedge ST(\psi, y, j)[x := w, i := S]$
- $(\mathcal{M})' \models ST(\overleftarrow{\Diamond}\psi, x, i)[x := w, i := S]$, by definition of ST.

By (1) and (2) for $S = \emptyset$ Property $\otimes$ follows, completing the proof. □

Now, we prove that if a first-order model satisfies the translation of an $\mathcal{SL}$ formula, then there exists an $\mathcal{SL}$ model that satisfies the original formula. To do so, let us introduce a translation from models of $2\mathcal{FOL}$ to models of $\mathcal{SL}$.

Definition 8. *Let $\mathcal{M} = \langle\{D, I\}, \mathcal{I}\rangle$ be a $2\mathcal{FOL}$ model and let $S \in I$, define $(\mathcal{M}, S)' = \langle W, R, V\rangle$ where*

- $W = D$
- $V(p) = \mathcal{I}(P_p)$
- $R = \{(w, v) \in W^2 \mid (w, S, v) \in \mathcal{I}(R')\}$.

The translation $(\ )'$ selects one part of the static model and turns it into an $\mathcal{SL}$ model. As we will now see, the selected part is the model variant where the formula is to be evaluated.

Theorem 4. *Let φ an $\mathcal{SL}$ formula. If $ST'(\varphi, x, i)$ is satisfiable, then φ is also satisfiable.*

Proof. We will prove the following. Let $\mathcal{M} = \langle \{D, I\}, \mathcal{I}\rangle$ be a $2\mathcal{FOL}$ model, and $S \in I$. Then

$$\mathcal{M} \models ST'(\varphi, x, i)[x := w; i := S] \text{ iff } (\mathcal{M}, S)', w \models \varphi.$$

Trivially, $\mathcal{M} \models Cond_1 \wedge Cond_2$. It remains to be proved by structural induction on the formula φ that:

$$\mathcal{M} \models ST(\varphi, x, i)[x := w; i := S] \text{ iff } (\mathcal{M}, S)', w \models \varphi.$$

In the proof below we only describe the left-to-right implication. The right-to-left implication can be proved following the steps in the converse direction.

$\varphi = p$: trivial by definition of $(\ )'$.
$\varphi = \neg\psi$, $\psi_1 \wedge \psi_2$: trivial.
$\varphi = \Diamond\psi$: Suppose $\mathcal{M} \models ST(\Diamond\psi, x, i)[x := w; i := S]$. We have:
- $\mathcal{M} \models \exists_{y:W}.R'(x, i, y) \wedge ST(\psi, y, i)[x := w; i := S]$
- $\mathcal{M} \models R'(x, i, y) \wedge ST(\psi, y, i)[x := w; i := S; y := v]$, for some new v, by definition of $\models$ $\otimes$
- $(w, S, v) \in \mathcal{I}(R')$, by definition of $\mathcal{I}$
- $(w, v) \in R$, by definition of $(\ )'$
- $(\mathcal{M}, S)', v \models \psi$, by I.H. on $\otimes$
- $(\mathcal{M}, S)', w \models \Diamond\psi$, from the last two items by definition of $\models$

$\varphi = \overleftarrow{\Diamond}\psi$: Suppose $\mathcal{M} \models ST(\overleftarrow{\Diamond}\psi, x, i)[x := w; i := S]$. We have:
- $\mathcal{M} \models \exists_{y:W,j:I}.S'(x, i, y, j) \wedge ST(\psi, y, j)[x := w; i := S]$
- $\mathcal{M} \models S'(x, i, y, j) \wedge ST(\psi, y, j)[x := w; i := S; y := v; j := U]$, for some new v and U, by definition of $\models$
- $\mathcal{M} \models R'(x, i, y)[x := w; i := S; y := v]$, by $Cond_1$ (right-to-left)
- $(w, S, v) \in \mathcal{I}(R')$, by definition of $\models$
- $(w, v) \in R$, by definition of $(\ )'$ $\otimes_1$
- $(\mathcal{M}, U)', v \models \psi$, by I.H. $\otimes_2$

Now let us prove that $(\mathcal{M}, U)' = (\mathcal{M}, S)'^*_{vw}$:
- Let $(\mathcal{M}, U)' = \langle W_1, R_1, V_1\rangle$, $(\mathcal{M}, S)'^*_{vw} = \langle W_2, R_2, V_2\rangle^*_{vw} = \langle W_2, R^*_{2\,vw}, V_2\rangle$
- Trivially $W_1 = W_2$ and $V_1 = V_2$.
- Now, we have $\mathcal{M} \models S'(x, i, y, j)[x{:=}w; i{:=} S; y{:=} v; j{:=}U]$.
  By $Cond_2$, this is equivalent to:
  $\mathcal{M} \models R'(w, i, v) \wedge R'(v, j, w) \wedge\ (w \neq v \ \rightarrow\ \neg R'(w, j, v))$
  $\wedge\forall_{w',v':W}\,.\,((w, v){\neq}(w', v')\wedge(w, v){\neq}(v', w')\ \rightarrow\ R'(w', i, v'){\leftrightarrow}R'(w', j, v'))$
  Now by definition of $(\ )'$, the above two facts give us $R_1 = R^*_{2\,vw}$.

We continue the proof knowing that $(\mathcal{M}, U)' = (\mathcal{M}, S)'^*_{vw}$:
- $\exists v \in W$ s.t. $(w, v) \in R$ and $(\mathcal{M}, S)'^*_{vw}, v \models \psi$, from $\otimes_1$ and $\otimes_2$
- $(\mathcal{M}, S)', w \models \overleftarrow{\Diamond}\psi$, by definition of $\models$. $\square$

The last two theorems provide the desired result:

Corollary 1. *Let φ an $\mathcal{SL}$ formula. φ is satisfiable if, and only if, $ST'(\varphi, x, i)$ is satisfiable.*

Sorts are a convenient, but non-essential, extension of first-order logic. It is indeed possible to translate many-sorted $\mathcal{FOL}$ to unsorted $\mathcal{FOL}$ following [10]. Hence $\mathcal{SL}$ is a fragment of unsorted first-order logic.

This translation is not equivalence preserving, since it does not preserve truth of $\mathcal{SL}$ formulas in the same models. However, using the construction of Definition 8, it is possible to select a part of a the $2\mathcal{FOL}$ model that satisfies the translation of a $\mathcal{SL}$ formula, to turn it into a $\mathcal{SL}$ model for the initial formula. This means we are able to build models for satisfiable formulas of $\mathcal{SL}$, granted we can build models for $2\mathcal{FOL}$ formulas.

4 Swap Logic as a Fragment of Hybrid Logic

Despite its dynamic behavior, $\mathcal{SL}$ remains a modal language: it retains a local perspective on models. It would be interesting to see if $\mathcal{SL}$ is a fragment of some known, probably very expressive, modal logic. It turns out that $\mathcal{SL}$ can be translated, maintaining truth on the same models, to the hybrid logic $\mathcal{H}(:,\downarrow)$.

Let us introduce formally the hybrid logic $\mathcal{H}(:,\downarrow)$ [7,5]. The syntax and semantics of this language are an extension of the basic modal logic $\mathcal{BML}$. Hybrid logics involve a special set of propositional symbols called *nominals*, that act as names pointing to a unique state in the model.

$\mathcal{H}(:,\downarrow)$ introduces the operators : and $\downarrow$. : is called the *satisfaction operator* and the formula $n{:}\varphi$ states that φ is true at the unique state where n holds. The *down-arrow binder* $\downarrow$ binds a given nominal to the current state in the model. Hence, $\downarrow n.\varphi$ intuitively means "after naming the current state n, φ holds". $\mathcal{H}(:,\downarrow)$ is more expressive than $\mathcal{BML}$. In fact, it is a reduction class of first order logic [5].

Definition 9. *Let the signature* $\langle \mathsf{PROP}, \mathsf{NOM}\rangle$ *be given, with* $\mathsf{NOM} \subseteq \mathsf{PROP}$. *The set* FORM *of formulas of* $\mathcal{H}(:,\downarrow)$ *over* $\langle \mathsf{PROP}, \mathsf{NOM}\rangle$ *is defined as:*

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

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

Syntactically, nominals can appear at the same places as regular propositional symbols, but they can also appear as parameters of the operators : and $\downarrow$. Models of hybrid logics are similar to models of $\mathcal{BML}$. However they need to make each nominal point to exactly one state. This can be done associating to each hybrid model an assignment function for nominals.

Definition 10. *A hybrid model* $\mathcal{M}$ *is a triple* $\langle W, R, V\rangle$ *where* W *is non empty,* $R \subseteq W \times W$ *and* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *is a valuation. An assignment* g *for* $\mathcal{M}$ *is an mapping* $g : \mathsf{NOM} \to W$. *Let* $\mathcal{M}$ *be a hybrid model,* w *a state in* $\mathcal{M}$, *and* g

an assignment, the semantics is defined as:

$$\begin{aligned}
\mathcal{M}, w \models p[g] &\text{ iff } w \in V(p), \text{ for } p \in \mathsf{PROP} \setminus \mathsf{NOM} \\
\mathcal{M}, w \models n[g] &\text{ iff } w = g(n), \text{ for } n \in \mathsf{NOM} \\
\mathcal{M}, w \models \neg\varphi[g] &\text{ iff } \mathcal{M}, w \not\models \varphi[g] \\
\mathcal{M}, w \models \varphi \wedge \psi[g] &\text{ iff } \mathcal{M}, w \models \varphi[g] \text{ and } \mathcal{M}, w \models \psi[g] \\
\mathcal{M}, w \models \Diamond\varphi[g] &\text{ iff for some } v \in W \text{ s.t. } (w, v) \in R, \mathcal{M}, v \models \varphi[g] \\
\mathcal{M}, w \models n{:}\varphi[g] &\text{ iff } \mathcal{M}, g(n) \models \varphi[g] \\
\mathcal{M}, w \models {\downarrow}n.\varphi[g] &\text{ iff } \mathcal{M}, w \models \varphi[g^n_w].
\end{aligned}$$

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

Now let us consider an equivalence preserving translation from $\mathcal{SL}$ to $\mathcal{H}(:, \downarrow)$. $\mathcal{SL}$ is a logic that is able to modify the relation of a model. On the other hand, $\mathcal{H}(:, \downarrow)$ can only update the assignment. Hence, to represent the model variant where some part of an $\mathcal{SL}$ formula is evaluated, we will take advantage of the binder $\downarrow$ to name the pair of points of the model where an edge should be considered as swapped. To ensure that formulas of the shape $\Diamond\varphi$ and $\overleftarrow{\Diamond}\varphi$ are satisfied in the correct model variant, the translation records the set of swapped edges as pairs of nominals in the set N. The translation also uses C and D subsets of N, such that the current state of evaluation is the source of all edges in C, and is the destination of all edges in D.

We now introduce the formal, quite involved, definition of the translation. Afterwards we will explain in detail how the different parts work together.

Definition 11. *Let $N \subseteq \mathsf{NOM} \times \mathsf{NOM}$. For $C, D \subseteq N$, define:*

$$\begin{aligned}
\mathsf{locate}(C, D, N) &= \bigwedge_{xy \in C} x \wedge \bigwedge_{xy \in N \setminus C} \neg x \wedge \bigwedge_{xy \in D} y \wedge \bigwedge_{xy \in N \setminus D} \neg y \\
\mathsf{forbid}(C, D) &= \bigwedge_{xy \in D} \neg x \wedge \bigwedge_{xy \in C} \neg y \\
\mathsf{pass}(C, N, \varphi) &= \bigvee_{xy \in C} y{:}(\varphi)'_N \\
\mathsf{swap}(C, N, \varphi) &= \bigvee_{xy \in C} y{:}(\varphi)'_{N \setminus xy \cup yx}.
\end{aligned}$$

Define $(\ \ )'_N$ from formulas of $\mathcal{SL}$ to formulas of $\mathcal{H}(:, \downarrow)$ as

$$\begin{aligned}
(p)'_N &= p \\
(\neg\varphi)'_N &= \neg(\varphi)'_N \\
(\psi \wedge \varphi)'_N &= (\psi)'_N \wedge (\varphi)'_N
\end{aligned}$$

$$(\Diamond\varphi)'_N = \bigwedge_{C, D \subseteq N} \mathsf{locate}(C, D, N) \to [\ \Diamond(\mathsf{forbid}(C, D) \wedge (\varphi)'_N)\ \vee \mathsf{pass}(C, N, \varphi)\ ]$$

$$\begin{aligned}
(\overleftarrow{\Diamond}\varphi)'_N = \quad & {\downarrow}x'.\ \Diamond\ (x'\ \wedge\ (\varphi)'_N) \\
& \vee \bigwedge_{C, D \subseteq N} \mathsf{locate}(C, D, N) \\
& \qquad \to [{\downarrow}x'.\Diamond{\downarrow}y'.(\neg x' \wedge \mathsf{forbid}(C, D) \wedge (\varphi)'_{N \cup y'x'})\ \vee\ \mathsf{swap}(C, N, \varphi)\ ]
\end{aligned}$$

where x' and y' are nominals that do not appear in N.

We include new edges in N only when we simulate a swap, that is, in the translation of formulas $\overleftarrow{\Diamond}$. We ensure this set never refers twice to the same edge, and it does not contain symmetric or reflexive edges. More specifically, the translation prevents $\Diamond$ and $\overleftarrow{\Diamond}$ formulas to be satisfied by "building again" an edge that has already been swapped. This is why we use locate and forbid simultaneously. On the other hand, it enables $\Diamond$ and $\overleftarrow{\Diamond}$ formulas to be satisfied by traversing a swapped edge, granted it is in the adequate direction in the current model variant. This is what pass and swap do. Furthermore swap ensures that N remains antisymmetric. Initially, N is empty, as we are in the model variant with no swapped edges. In that case, the translation is simplified to:

$$(\Diamond\varphi)'_\emptyset = \Diamond(\varphi)'_\emptyset$$
$$(\overleftarrow{\Diamond}\varphi)'_\emptyset = \downarrow x'.\Diamond(x' \wedge (\varphi)'_\emptyset) \quad \vee \quad \downarrow x'.\Diamond\downarrow y'.(\neg x' \wedge (\varphi)'_{y'x'})$$

In $(\overleftarrow{\Diamond}\varphi)'_\emptyset$ we see two occurrences of $(\varphi)'_N$ (for some N). Hence the translation may produce formulas of size exponential in terms of the size of the initial formula, for instance, $(\overleftarrow{\Diamond}\overleftarrow{\Diamond}\ldots\overleftarrow{\Diamond}p)'_\emptyset$.

Now we show that the translation maintains equivalence. To do so, we show by inductive argument that the set N really corresponds to swapped edges. Observe that there are no free nominals in the formulas produced by the translation. This means that such formulas are satisfiable in models where the assignment for the nominals of the language does not matter.

Theorem 5. *Given φ a formula of $\mathcal{SL}$, $\mathcal{M}, w$ a pointed model and g an arbitrary assignment. Then:*

$$\mathcal{M}, w \models \varphi \quad \textit{iff} \quad \mathcal{M}, w \models (\varphi)'_\emptyset[g].$$

Proof. Let $\mathcal{M}, w$ a pointed model. Let us show by induction that for all $N \subseteq \mathsf{NOM} \times \mathsf{NOM}$, for all $S \subseteq W \times W$, and for all $g : \mathsf{NOM} \to W$ such that:

(Props)
- N, S are irreflexive and antisymmetric.
- If $xy \in N$ then $g(x)g(y) \in S$.
- If $wv \in S$, then there exists $x, y \in N$ such that $g(x)g(y) = wv$.
- If $xy, x'y' \in N$, then $g(x)g(y) \neq g(x')g(y')$.

then:

$$\mathcal{M}^*_S, w \models \varphi \quad \text{iff} \quad \mathcal{M}, w \models (\varphi)'_N[g].$$

Where $\mathcal{M}^*_S$ is the generalization of $\mathcal{M}^*_{vw}$ of Definition 2 to sets of pairs. $\mathcal{M}^*_S$ is uniquely defined since we assume S to be irreflexive and asymmetric.
First, observe that the following can be easily verified:

(1) N,S and g comply with (Props) if, and only if, $N \cup y'x'$ (for $x'y', y'x' \notin N$), $S \cup vw$ (for $wv, vw \notin S$) and $(g^{x'}_w)^{y'}_v$ comply with (Props).
(2) $N \cup xy$, $S \cup wv$ and g comply with (Props) if, and only if, $N \cup yx$, $S \cup vw$ and g comply with (Props).

Now let us examine the two non-trivial inductive cases:

$\boldsymbol{\varphi = \Diamond\psi}$**:** we have the following successive equivalences:
- $\mathcal{M}^*_S, w \models \Diamond\psi$
- exists $v \in W$ such that $wv \in R^*_S$ and $\mathcal{M}^*_S, v \models \psi$
- exists $v \in W$ such that either $wv \in R \setminus S$ and $\mathcal{M}^*_S, v \models \psi$, or $wv \in S$ and $\mathcal{M}^*_S, v \models \psi$
- exists $v \in W$ such that either $wv \in R \setminus S$ and $\mathcal{M}, v \models (\psi)'_N[g]$, or $wv \in S$ and $\mathcal{M}, v \models (\psi)'_N[g]$ (by IH)
- either $\mathcal{M}, w \models \bigwedge_{C,D \subseteq N} \mathsf{locate}(C, D, N) \to \Diamond(\mathsf{forbid}(C, D) \wedge (\varphi)'_N)[g]$ or $\mathcal{M}, w \models \bigwedge_{C,D \subseteq N} \mathsf{locate}(C, D, N) \to \mathsf{pass}(C, N, \varphi)[g]$
- $\mathcal{M}, w \models (\Diamond\varphi)'_N[g]$.

$\boldsymbol{\varphi = \overleftarrow{\Diamond}\psi}$**:** we have the following successive equivalences:
- $\mathcal{M}^*_S, w \models \overleftarrow{\Diamond}\psi$
- exists $v \in W$ s.t. $wv \in R^*_S$ and $(\mathcal{M}^*_S)^*_{vw}, v \models \psi$
- either: $ww \in R^*_S$ and $\mathcal{M}^*_S, w \models \psi$
  or: exists $v \neq w$ s.t. $wv \in (R \setminus S)$ and $(\mathcal{M}^*_S)^*_{vw}, v \models \psi$, i.e., $\mathcal{M}^*_{S \cup vw}, v \models \psi$
  or: exists $v \neq w$ s.t. $wv \in S$ and $(\mathcal{M}^*_S)^*_{vw}, v \models \psi$, i.e., $\mathcal{M}^*_{S \setminus wv \cup vw}, v \models \psi$
- either: $ww \in R^*_S$ and $\mathcal{M}, w \models (\psi)'_N[g]$ (IH)
  or: exists $v \neq w$ s.t. $wv \in (R \setminus S)$ and $\mathcal{M}, v \models (\psi)'_{N \cup y'x'}[(g^{x'}_w)^{y'}_v]$ (IH, **(1)**)
  or: exists $v \neq w$ s.t. $wv \in S$ and $\mathcal{M}, v \models (\psi)'_{N \setminus xy \cup yx}[g]$ (IH, **(2)**)
- either: $\mathcal{M}, w \models \downarrow x'.\Diamond(x' \wedge (\varphi)'_N)[g]$
  or: $\mathcal{M}, w \models \bigwedge_{C,D \subseteq N} \mathsf{locate}(C, D, N) \to \downarrow x'.\Diamond\downarrow y'.(\neg x' \wedge \mathsf{forbid}(C, D) \wedge (\varphi)'_{N \cup y'x'})[g]$
  or: $\mathcal{M}, w \models \bigwedge_{C,D \subseteq N} \mathsf{locate}(C, D, N) \to \mathsf{swap}(C, N, \varphi)[g]$
- $\mathcal{M}, w \models (\overleftarrow{\Diamond}\varphi)'_N[g]$.

We proved that for all N, S, g such that Props hold, we have $\mathcal{M}^*_S, w \models \varphi$ if, and only if, $\mathcal{M}, w \models (\varphi)'_N[g]$. It suffices to take the particular case where N, S and g are empty, to obtain the result we wanted. □

This translation is anything but economic: it may produce formulas of size exponential in function of the input formula. Its advantage with respect to the translation to $2\mathcal{FOL}$ of Section 3, is that it is *equivalence preserving*. That is, it works on the same models as $\mathcal{SL}$. This has at least two nice consequences. First, to build a model for a $\mathcal{SL}$ formula, it is enough to build a model for the translation. Second, equivalence preserving intuitively means that hybrid logic is at least as expressive as swap logic (but, possibly, not as concise). Does the inverse hold? To answer this question, we will need a notion of bisimulation for $\mathcal{SL}$, as we will see in the next section.

5 Bisimulation and Expressive Power

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. This will not suffice for $\mathcal{SL}$ where we also need to capture the dynamic behaviour of the $\overleftarrow{\Diamond}$ operator. The proper notion of $\mathcal{SL}$-bisimulations links states together with the current accessibility relation.

Definition 12 ($\mathcal{SL}$-Bisimulations). *Let $\mathcal{M} = \langle W, R, V\rangle$, $\mathcal{M}' = \langle W', R', V'\rangle$ be two models. A non empty relation $Z \subseteq (W \times \mathcal{P}(W^2)) \times (W' \times \mathcal{P}(W'^2))$ is an $\mathcal{SL}$-bisimulation if it satisfies the following conditions. If $(w, S)Z(w', S')$ then*

(Atomic Harmony) *for all $p \in \mathsf{PROP}$, $\mathcal{M}, w \models p$ iff $\mathcal{M}', w' \models p$;*
(Zig) *if wSv then for some v', $w'S'v'$ and $(v, S)Z(v', S')$;*
(Zag) *if $w'S'v'$ then for some v, wSv and $(v, S)Z(v', S')$;*
(S-Zig) *if wSv then for some v', $w'S'v'$ and $(v, S^*_{vw})Z(v'S'^*_{v'w'})$;*
(S-Zag) *if $w'S'v'$ then for some v, wsv and $(v, S^*_{vw})Z(v'S'^*_{v'w'})$.*

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

Theorem 6. *Let $\mathcal{M} = \langle W, R, V\rangle$ and $\mathcal{M}' = \langle W', R', V'\rangle$ be two models, $w \in W$, $w' \in W'$, and let $S \subseteq W^2, S' \subseteq W'^2$. If there is an $\mathcal{SL}$-bisimulation Z between $\mathcal{M}, w$ and $\mathcal{M}', w'$ such that $(w, S)Z(w', S')$ then for any formula $\varphi \in \mathcal{SL}$, $\langle W, S, V\rangle, w \models \varphi$ iff $\langle W', S', V'\rangle, w' \models \varphi$.*

Proof. The proof is by structural induction on $\mathcal{SL}$ formulas. The base case holds by Atomic Harmony, and the $\wedge$ and $\neg$ cases are trivial.

$\varphi = \Diamond\psi$: Suppose $\langle W, S, V\rangle, w \models \Diamond\psi$. Then there is v in W s.t. wSv 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.

$\varphi = \overleftarrow{\Diamond}\psi$: For the left to the right direction suppose $\langle W, S, V\rangle, w \models \overleftarrow{\Diamond}\psi$. Then there is v in W such that wSv and $\langle W, S^*_{vw}, V\rangle, v \models \psi$. By S-Zig we have v' in W' s.t. $w'S'v'$ 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 $\langle W', S', V'\rangle, w' \models \overleftarrow{\Diamond}\psi$. For the other direction use S-Zag.□

Example 4. The two models of Figure 2 are $\mathcal{SL}$-bisimilar.

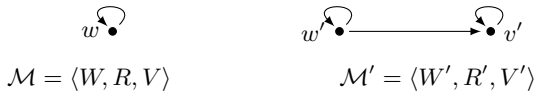


Fig. 2. Two $\mathcal{SL}$-bisimilar models.

The $\mathcal{SL}$-bisimulation between $\mathcal{M}, w$ and $\mathcal{M}, w'$ is the set of pairs:

$$\begin{array}{l}(1)\ \{(w, R), (w', R'),\\ (2)\ (w, R),\ (v', R'),\\ (3)\ (w, R),\ (v', R'^*_{v'w'}),\\ (4)\ (w, R),\ (w', R'^*_{v'w'})\}\end{array}$$

(1) represents the starting models $\mathcal{M}, w$ and $\mathcal{M}, w'$. (2) is needed to satisfy the **Zag** condition on (1), (3) is needed to satisfy **S-Zag** on (1), and (4) is needed to satisfy **Zag** on (3). Note that the first element of all pairs never changes, given that swapping a reflexive edge has no effect.

Example 5. There is no $\mathcal{SL}$-bisimulation between the models of Figure 3.

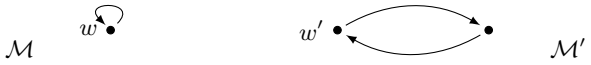


Fig. 3. Two non $\mathcal{SL}$-bisimilar models.

Indeed, the formula $\overleftarrow{\Diamond}\Diamond\Box\bot$ is satisfiable in $\mathcal{M}', w'$ and not in $\mathcal{M}, w$. Notice that the models are $\mathcal{BML}$-bisimilar.

We are now ready to investigate the expressive power of $\mathcal{SL}$.

Definition 13. *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 $\mathcal{L}'$ is strictly more expressive *than $\mathcal{L}$ (notation $\mathcal{L} < \mathcal{L}'$) if $\mathcal{L} \leq \mathcal{L}'$ but not $\mathcal{L}' \leq \mathcal{L}$.*

Our first result is fairly straightforward as it builds upon Example 5: $\mathcal{BML}$ is strictly less expressive than $\mathcal{SL}$.

Theorem 7. $\mathcal{BML} < \mathcal{SL}$.

Proof. We have to provide a translation from $\mathcal{BML}$ formulas to $\mathcal{SL}$. This is trivial as $\mathcal{BML}$ is a fragment of $\mathcal{SL}$. To prove $\mathcal{SL} \not\leq \mathcal{BML}$ we show two models that are bisimilar in $\mathcal{BML}$ and a $\mathcal{SL}$ formula that distinguishes them. Consider the models of Figure 3. They are bisimilar for $\mathcal{BML}$ but the $\mathcal{SL}$ formula $\overleftarrow{\Diamond}\Diamond\Box\bot$ distinguishes them. □

The second result is a comparison between $\mathcal{SL}$ and $\mathcal{H}(:,\downarrow)$:

Theorem 8. $\mathcal{SL} < \mathcal{H}(:,\downarrow)$.

Proof. The translation $(\ )'_{\emptyset}$ of Definition 11 is equivalence-preserving from $\mathcal{SL}$ to $\mathcal{H}(:,\downarrow)$. The models in Example 4 (Figure 2) are $\mathcal{SL}$-bisimilar. The $\mathcal{H}(:,\downarrow)$ formula $\downarrow x.\Diamond\neg x$ distinguishes them, being true in $\mathcal{M}', w'$ and false in $\mathcal{M}, w$. □

Let us now compare $\mathcal{SL}$ with a family of dynamic modal logics called *memory logics*. Memory logics [1,14] are modal logics with the ability to *store* the current state of evaluation into a set, and to consult whether the current state of evaluation belongs to this set.

Definition 14 (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* $\varphi, \psi \in \mathsf{FORM}$.

Definition 15 (Semantics of memory logics). *A model* $\mathcal{M} = \langle W, R, V, S\rangle$ *is an extension of an* $\mathcal{SL}$ *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 $\mathcal{ML}(ⓡ, ⓚ)$ [3,14]. We now show that the expressive power of $\mathcal{SL}$ is uncomparable with both $\mathcal{ML}(ⓡ, ⓚ)$ and $\mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$.

Theorem 9. $\mathcal{SL} \not\geq \mathcal{ML}(\langle\!\langle \mathrm{r} \rangle\!\rangle, ⓚ)$.

Proof. As we showed before, no $\mathcal{SL}$ formula can distinguish models of Figure 2, but the formula $\langle\!\langle \mathrm{r} \rangle\!\rangle\neg ⓚ$ does (being satisfiable in $\mathcal{M}', w'$ but not in $\mathcal{M}, w$). □

Theorem 10. $\mathcal{ML}(ⓡ, ⓚ) \not\geq \mathcal{SL}$.

Proof. Models on Figure 4 are bisimilar in $\mathcal{ML}(ⓡ, ⓚ)$. Indeed they are bisimilar for the basic modal logic and acyclic, hence $\mathcal{ML}(ⓡ, ⓚ)$ cannot see again points of the model stored using ⓡ, which means ⓚ is always false after taking an accessibility relation. The formula $\overleftarrow{\Diamond}\Diamond\Diamond\top$ is true in $\mathcal{M}, w$ but not in $\mathcal{M}', w'$. □

Fig. 4. Two $\mathcal{ML}(ⓡ, ⓚ)$-bisimilar models.

From Theorems 9 and 10 we can conclude that the expressive powers of $\mathcal{ML}(ⓡ, ⓚ)$ and $\mathcal{SL}$ are uncomparable. The same holds for $\mathcal{ML}(\langle\langle \mathrm{r} \rangle\rangle, ⓚ)$ and $\mathcal{SL}$. Figure 5 sums up the results of this section (arrows that are not results of this paper, have been extracted from [3]).

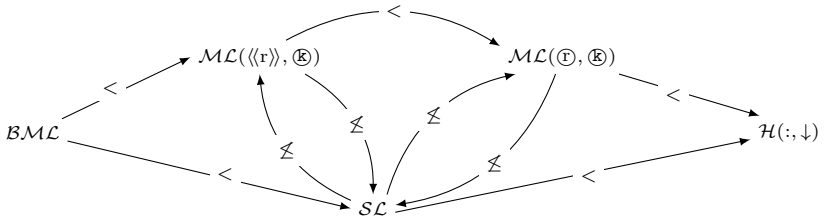


Fig. 5. Comparison of expressive power.

In [2] other operators than can modify the accessibility relation of a model have been investigated beside $\mathcal{SL}$: in addition to the classic sabotage operator, we introduced a *local* version of sabotage logic in which the operator deletes edges as it traverses them to reach accessible states, and a "bridge" operator that can move to an unreachable state in the model while making it a successor of the current state of evaluation. Appropriate notions of bisimulation for these logics have been introduced, together with a proof that their expressive powers are all uncomparable.

6 Complexity of Model Checking and Satisfiability

In this section we examine the computational behaviour of $\mathcal{SL}$ with regards to two common tasks: model checking and satisfiability. At this point of our investigation, we have an idea of what we can expect. Indeed, we know that these tasks have to be as easy as for the basic modal logic, and as hard as for first-order logic. Model checking of first-order is PSpace-complete [9,18,21], while for the basic modal logic is in P [11]. Here we provide a proof that model checking a formula of $\mathcal{SL}$ is PSpace-complete.

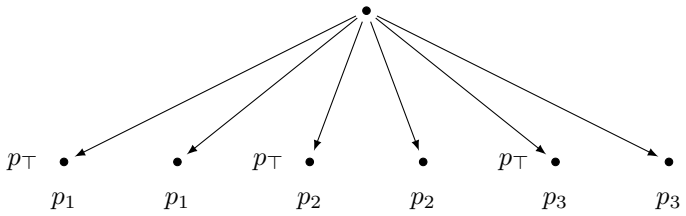
Theorem 11. *Given an arbitrary finite model $\mathcal{M}$ and φ a formula in $\mathcal{SL}$, deciding whether φ is satisfied by some state of $\mathcal{M}$ is PSpace-complete.*

Proof. A Quantified Boolean Formula (QBF) [15] is a Boolean formula with universal or existential quantification on its variables, e.g. $\forall x_1.\exists x_2.((x_1 \vee x_2) \wedge (x_1 \vee \exists x_3.\neg x_3))$. Truth of a QBF under some assignment v is defined as for propositional logic except for the following cases: a formula of the form $\forall x.\varphi$ is true under a valuation v iff φ is true according to the valuation $v[x := 0]$ *and* it is true according to $v[x := 1]$; $\exists x.\varphi$ is true under v iff it is true under $v[x := 0]$ *or* it is true under $v[x := 1]$. For instance, $\forall x_1.\exists x_2.(x_1 \vee x_2)$ is true under any valuation, while $\forall x_1.\forall x_2.(x_1 \vee x_2)$ is false under any valuation. In such cases we simply say that these formulas are respectively true and false.

Determining whether a QBF is true is a PSpace-hard problem. We will reduce this problem to the model checking problem of $\mathcal{SL}$. Let α be a QBF with variables $\{x_1, \ldots, x_k\}$. Without loss of generality, we assume that all its variables are quantified, and that they are quantified only once. 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:

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

For instance, $\mathcal{M}_3$ is as follows:



Let $(\ )'$ be the following linear translation from QBF to $\mathcal{SL}$

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

The evaluation of the $\mathcal{SL}$ formula in the model simulates the assignments of Boolean variables of the QBF α by swapping around edges between w and its successors. A variable x_i is assigned to 1 if w_i^1 is no longer accessible from w, and it is assigned to 0 otherwise.

It remains to prove that α is true iff $\mathcal{M}_k, w \models (\alpha)'$. For a model $\mathcal{M}$ with relation R we define $v_R : \{x_1, \ldots, x_k\} \to \{0, 1\}$ as "$v_R(x_i) = 1$ iff $(w, w_i^1) \notin R$",

in the present case, iff the edge between w and w_i^1 has been swapped. We write $v \models_{\text{qbf}} \beta$ if valuation $v : \{x_1, \ldots, x_k\} \to \{0, 1\}$ satisfies the QBF β.

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 true.

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 $(v_R[x_i := 0] \models_{\text{qbf}} \gamma$ or $v_R[x_i := 1] \models_{\text{qbf}} \gamma)$ 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 iff $(\mathcal{M}^*_{w_i^0 w}, w_i^0 \models \Diamond(\gamma)'$ or $\mathcal{M}^*_{w_i^1 w}, w_i^1 \models \Diamond(\gamma)')$ iff $\mathcal{M}, w \models \overleftarrow{\Diamond}(p_i \wedge \Diamond(\gamma)')$ iff $\mathcal{M}, w \models (\exists x_i.\gamma)'$.

This shows that the model checking problem of $\mathcal{SL}$ is PSpace-hard. To prove that it is in PSpace, reason as follows. 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 of the form $\overleftarrow{\Diamond}\varphi$ that allocate twice as much 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 swapping operators $\overleftarrow{\Diamond}$ and $\overleftarrow{\Box}$ of the input formula. By proceeding depth-first, the algorithm runs using only polynomial space in the size of the model. □

For a modal language with just one, syntactically simple, extra operator, $\mathcal{SL}$ has a surprisingly hard model checking problem. Why is it that adding the $\overleftarrow{\Diamond}$ operator makes the difficulty of this task jump to the same as first-order logic? We saw that this is happening because the language can now *store* information in the model (by swapping edges) and then *check* again if some piece of information is present or not (just using the usual modal operators). This is different from model checking for the basic modal logic, where the model against which one checks some formula remains static during all the evaluation.

We now turn to the satisfiability problem. Again, our expectations lie between PSpace-completeness for the basic modal logic, and undecidability as it is the case for first-order logic, the hybrid logic $\mathcal{H}(:, \downarrow)$ [5] and most memory logics [3]. We are indeed going to see that we can reduce the satisfiability problem of $\mathcal{ML}(ⓡ, ⓚ)$ to the one of $\mathcal{SL}$, hence showing its undecidability.

We need to simulate the behaviour of $\mathcal{ML}(ⓡ, ⓚ)$, i.e., the ability to memorize and check states, without having an external memory. What we have instead is the ability to swap edges in the model. We will build models that contain *switches*, special edges whose position – "off" by default, and "on" if the direc-

tion of the edge has been swapped around – will represent whether a point of the model has been remembered with the ⓡ operator. We will simulate the ⓚ predicate by querying the position of these switches.

Let us introduce the translation from $\mathcal{ML}(ⓡ, ⓚ)$ to $\mathcal{SL}$:

Definition 16. *Let φ be a formula of $\mathcal{ML}(ⓡ, ⓚ)$ that does not contain the propositional symbols s and sw. Let $\mathsf{Tr}(\varphi)$ be the following formula:*

$$
\begin{array}{ll}
(1) & \quad s \ \wedge \Box^{(4)} \neg s \ \wedge \ \neg sw \ \wedge \ \Box \neg sw \\
(2) & \wedge \ \Box(\ \Diamond(sw \wedge \Box\bot) \ \wedge \ \overleftarrow{\Box}(sw \rightarrow \Box\neg\Diamond sw)\ ) \\
(3) & \wedge \ \Box\Box(\ \neg sw \ \rightarrow \ \Diamond(sw \wedge \Box\bot) \ \wedge \ \overleftarrow{\Box}(sw \rightarrow \Box\neg\Diamond sw)\ ) \\
(4) & \wedge \ \overleftarrow{\Box}\overleftarrow{\Box}(\ \neg s \wedge \neg sw \ \rightarrow \overleftarrow{\Diamond}(sw \wedge \Diamond\Diamond\Diamond(s \wedge \Diamond\neg\Diamond sw))\ ) \\
(5) & \wedge \ \overleftarrow{\Box}\overleftarrow{\Box}\Box\Box\Box\Box(sw \rightarrow \Box\bot) \\
(6) & \wedge \ \Diamond(\varphi)'.
\end{array}
$$

With $(\ \ )'$ defined as:

$$
\begin{array}{ll}
(ⓡ\psi)' & = (\ \Diamond sw \ \rightarrow \ \overleftarrow{\Diamond}(sw \wedge \Diamond(\psi)')\ ) \quad \wedge \quad (\ \neg\Diamond sw \ \rightarrow \ (\psi)'\ ) \\
ⓚ' & = \neg\Diamond sw \\
(\psi \otimes \chi)' & = (\psi)' \otimes (\chi)' \quad \textit{for} \otimes \in \{\vee, \wedge\} \\
(\neg\psi)' & = \neg(\psi)' \\
(\Diamond\psi)' & = \Diamond(\neg sw \ \wedge \ (\psi)') \\
(\Box\psi)' & = \Box(\neg sw \ \rightarrow \ (\psi)')
\end{array}
$$

In $\mathsf{Tr}(\varphi)$, the propositional symbol s is used to refer to the evaluation point, that will also be a spy point, i.e., a point that has direct access to all the points in the part of the model that represents the model of the original memory logic formula. sw will be true at "switch points".

(1) puts adequate constraints on the truth of propositional symbols s and sw. (2) and (3) initialize exactly one switch for every point accessible in one or two steps from the evaluation point. (4) links the evaluation point to every point of the connected component except itself and the switch points.

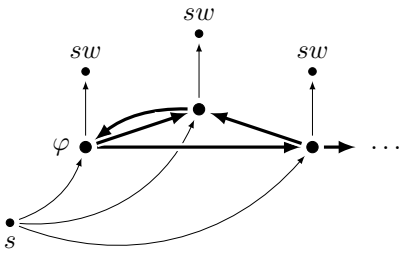
(5) ensures that switch points can be reached from the evaluation point by a unique path. Indeed, if this were not the case, then it would be possible to swap around two edges leading to some switch point, then come back to the evaluation point in two steps by this new path, and, by some other path,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

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

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

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

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



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

Theorem 12. *Let φ be a formula of $\mathcal{ML}(ⓡ, ⓚ)$ that does not contain the propositional symbols s and sw. Then, φ and $\mathsf{Tr}(\varphi)$ are equisatisfiable.*

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

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

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

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

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

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

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

We immediately get:

Theorem 13. *The satisfiability problem of $\mathcal{SL}$ is undecidable.*

7 Conclusions

In this paper we have extended the basic modal language with the $\overleftarrow{\Diamond}$ operator to describe dynamic aspects of relational models. $\overleftarrow{\Diamond}$ is a diamond-like operator that in addition has the ability to invert pairs of related elements in the domain while traversing an edge of the accessibility relation. The $\mathcal{SL}$ logic obtained by adding the $\overleftarrow{\Diamond}$ operator to the basic modal logic is very expressive: it has a PSpace-complete model checking problem and an undecidable satisfiability problem.

Other dynamic languages that can modify the model have been investigated in the literature (e.g., sabotage logics [19,12,13], memory logics [14,3,4], hybrid logics [7,5]), and we have discussed in detail the relation between some of these languages and $\mathcal{SL}$. In particular, we have introduced an adequate notion of bisimulation for $\mathcal{SL}$ and used it to show that the expressive power of $\mathcal{SL}$ lies strictly in between the expressive powers of the basic modal logic $\mathcal{BML}$ and the hybrid logic $\mathcal{H}(:,\downarrow)$, while it is uncomparable with the expressive powers of the memory logics $\mathcal{ML}(ⓡ, ⓚ)$ and $\mathcal{ML}(\langle\langle \mathrm{r} \rangle\rangle, ⓚ)$. We also investigated the expressive power of $\mathcal{SL}$ in relation to first-order logic. We have shown that despite the dynamic behaviour of $\mathcal{SL}$ it is indeed possible to capture $\mathcal{SL}$ as a fragment of first-order logic, giving a satisfiability preserving translation to two sorted first-order logic.

Many theoretical aspects of $\mathcal{SL}$ remain to be investigated. For example, it would be interesting to obtain an axiomatic characterization which is sound and complete. The task is probably non trivial, as the logic fails to be closed under uniform substitution (for example, the formula $\Diamond p \leftrightarrow \overleftarrow{\Diamond} p$ is a validity, but if we substitute the propositional variable p by the formula $\Diamond p$ we can easily check that the equivalence is not preserved). A proper axiomatization will require an adequate definition of when a formula is free to substitute another formula in an axiom. Another challenge is to exploit the expressive power of these logics that are able to modify the accessibility relation, to encode Dynamic Epistemic Logics (DEL). This family of logics includes, for example, Public Announcement Logic [16,20], in which we can make some announcement that is public for all the involved agents, deleting all the states where such announcement does not hold. Another well known DEL is Action Model Logic [20]. In this case, some semantic objects called *action models* are introduced as part of modal operators, and formulas are evaluated in traditional epistemic models. Action models can be seen as epistemic actions, representing some information change in epistemic models. It would be interesting to represent these epistemic logics combining or modifying our model changing operators, for example using them to find a completely syntactic way to represent actions of the agents without the need to include semantic objects in modalities.

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 International Associé "INFINIS".

References

1. C. Areces. Hybrid logics: The old and the new. In X. Arrazola and J. Larrazabal, editors, *Proceedings of LogKCA-07*, pages 15–29, San Sebastian, Spain, 2007.
2. C. Areces, R. Fervari, and G. Hoffmann. Moving arrows and four model checking results. In *Proceedings of WoLLIC 2012*, Buenos Aires, Argentina, September 2012.
3. 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.
4. C. Areces, S. Figueira, and S. Mera. Completeness results for memory logics. *Annals of Pure and Applied Logic*, 163(7):961–972, 2012.
5. C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*, pages 821–868. Elsevier, 2006.
6. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2001.
7. P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995.
8. P. Blackburn and J. van Benthem. Modal logic: A semantic perspective. In *Handbook of Modal Logic*. Elsevier North-Holland, 2006.
9. A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of 9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.
10. H. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.
11. R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. The MIT Press, Cambridge, Massachusetts, 1995.
12. C. Löding and P. Rohde. Model checking and satisfiability for sabotage modal logic. In P. Pandya and J. Radhakrishnan, editors, *Proceedings of Foundations of Software Technology and Theoretical Computer Science, 23rd Conference*, volume 2914 of *Lecture Notes in Computer Science*, pages 302–313. Springer, 2003.
13. 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.
14. S. Mera. *Modal Memory Logics*. PhD thesis, Universidad de Buenos Aires Facultad de Ciencias Exactas y Naturales Departamento de Computación, and UFR STMIA - Ecole Doctorale IAEM Lorraine Département de Formation Doctorale en Informatique, 2009.
15. C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
16. J. Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.
17. P. Rohde. *On games and logics over dynamically changing structures*. PhD thesis, RWTH Aachen, 2006.
18. L. Stockmeyer. *The complexity of decision problems in automata theory and logic*. Ph. d. thesis, MIT, Cambridge, Mass., 1974.
19. J. van Benthem. An essay on sabotage and obstruction. In D. Hutter and W. Stephan, editors, *Mechanizing Mathematical Reasoning*, volume 2605 of *Lecture Notes in Computer Science*, pages 268–276. Springer, 2005.
20. H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Kluwer, 2007.
21. M. Vardi. The complexity of relational query languages. In H. Lewis, B. Simons, W. Burkhard, and L. Landweber, editors, *Symposium on Theory of Computing*, pages 137–146. ACM, 1982.

22. H. Wang. Logic of many-sorted theories. *Journal of Symbolic Logic*, 17(2):105–116, 1952.