Model Checking and Satisfiability for Sabotage Modal Logic

Christof Löding and Philipp Rohde

Lehrstuhl für Informatik VII, RWTH Aachen
{loeding,rohde}@informatik.rwth-aachen.de

Abstract. We consider the sabotage modal logic SML which was suggested by van Benthem. SML is the modal logic equipped with a 'transition-deleting' modality and hence a modal logic over changing models. It was shown that the problem of uniform model checking for this logic is PSPACE-complete. In this paper we show that, on the other hand, the formula complexity and the program complexity are linear, resp., polynomial time. Further we show that SML lacks nice model-theoretic properties such as bisimulation invariance, the tree model property, and the finite model property. Finally we show that the satisfiability problem for SML is undecidable. Therefore SML seems to be more related to FO than to usual modal logic.

1 Introduction

In [1] van Benthem considered 'sabotage modal logics' which are modal logics over changing models. He introduced a cross-model modality referring to submodels from which objects have been removed. SML is modal logic equipped with a 'transition-deleting' modality. This logic is capable of expressing changes of transition systems itself in contrast to the usual specifications for systems, where only properties of a static system are expressed. As an application one can consider computer or traffic networks where connections may break down. One can express problems related to this situation by first order specifications, but then one has to put up with the high complexity of FO. So SML seems to be a moderate strengthening of modal logic for this kind of problems. In Sec. 2 we repeat the formal definition of the sabotage modal logic SML which is interpreted over edge-labelled transition systems.

Two main questions arise in this context: the model checking problem and the synthesis problem for SML. The model checking problem is the question, given a transition system and a system specification expressed in SML, does the system satisfy the specification? The synthesis problem asks, given a system specification, whether there is a transition system which satisfies the specification. In [5] we showed that the problem of uniform model checking for SML is PSPACE-complete. But in many cases one of the inputs for the model checking problem is fixed, either a single property is specified by a formula and one wants to check it for several systems; or there are different properties which should

P.K. Pandya and J. Radhakrishnan (Eds.): FSTTCS 2003, LNCS 2914, pp. 302–313, 2003.

be verified for a single system. For modal and temporal logics these two views of the complexity are usually referred to as program complexity and formula complexity of the model checking problem. In Sec. 3 we show that the formula complexity for SML is linear in the size of the formula and the program complexity for SML is polynomial in the size of the transition system. This result is in contrast to many other logics like LTL and CTL* where the formula complexity is as hard as the combined model checking complexity (cf. [6]). On the other hand, this result constitutes an interesting advantage over first order logic, since model checking for FO with a fixed transition system is PSPACE-complete (cf. [3]). Before we deal with the synthesis problem, we show in Sec. 4 that SML, in contrast to modal logic, lacks nice model-theoretic properties such as bisimulation invariance, the tree model property, and the finite model property. In Sec. 5 we split the synthesis problem into three questions: given a system specification expressed as an SML-formula, the satisfiability problem asks whether there is a transition system at all which satisfies the specification, i.e. the system might be finite or infinite. The finite satisfiability problem asks for finite systems as models of the formula. And finally, the infinity axiom problem is the question whether a given formula has only infinite models. We will show that for SML all three problems are undecidable. We do that by reducing appropriate modifications of Post's correspondence problem to these problems.

We would like to thank Johan van Benthem for several ideas and comments on the topic.

2 Sabotage Modal Logic

In this section we repeat the formal definition of the sabotage modal logic SML with a 'transition-deleting' modality. We interpret the logic over edge-labelled transition systems. Let $\mathrm{Prop} = \{p, p', p'', \dots\}$ be a set of unary predicate symbols. A (finite) transition system $\mathcal{T}$ is a tuple (S, Σ, R, L) with a finite set of states S, a finite alphabet Σ, a ternary transition relation $R \subseteq S \times \Sigma \times S$ and a labelling function $L : S \to 2^{\mathrm{Prop}}$. Let $p \in \mathrm{Prop}$ and $a \in \Sigma$. Formulae of the *sabotage modal logic* SML over transition systems are inductively defined by the grammar

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond_a \varphi \mid \Diamond\!\!\!\!-_a \varphi.$$

As usual, $\bot$ is an abbreviation for $\neg\top$. The dual modalities are defined by $\Box_a\varphi := \neg\Diamond_a\neg\varphi$ and $\boxminus_a\varphi := \neg\Diamond\!\!\!\!-_a\neg\varphi$. Let $\mathcal{T} = (S, \Sigma, R, L)$ be a transition system. To define the semantics of SML and for later use we define the transition system $\mathcal{T}_E$ for a set $E \subseteq R$ as $\mathcal{T}_E := (S, \Sigma, R \setminus E, L)$.

For a given state $s \in S$ we define the semantics of SML inductively by

$$\begin{aligned}
(\mathcal{T}, s) &\models \top && \text{for all } \mathcal{T} \text{ and all } s \in S,\\
(\mathcal{T}, s) &\models p && \text{iff } p \in L(s),\\
(\mathcal{T}, s) &\models \neg\varphi && \text{iff not } (\mathcal{T}, s) \models \varphi,
\end{aligned}$$

$$
\begin{aligned}
(\mathcal{T}, s) &\models \varphi \vee \psi && \text{iff } (\mathcal{T}, s) \models \varphi \text{ or } (\mathcal{T}, s) \models \psi, \\
(\mathcal{T}, s) &\models \Diamond_a \varphi && \text{iff there is } s' \in S \text{ with } (s, a, s') \in R \text{ and } (\mathcal{T}, s') \models \varphi, \\
(\mathcal{T}, s) &\models \Diamond\!\!\!\!-_a \varphi && \text{iff there is } (t, a, t') \in R \text{ with } (\mathcal{T}_{\setminus\{(t,a,t')\}}, s) \models \varphi.
\end{aligned}
$$

A measure for the complexity of an SML-formula φ is the number of nested sabotage operators. We call this the *sabotage depth* $\mathrm{sd}(\varphi)$ of φ and define inductively

$$
\begin{aligned}
\mathrm{sd}(\top) := \mathrm{sd}(p) &:= 0, & \mathrm{sd}(\varphi_1 \vee \varphi_2) &:= \max\{\mathrm{sd}(\varphi_1), \mathrm{sd}(\varphi_2)\}, \\
\mathrm{sd}(\neg\psi) := \mathrm{sd}(\Diamond_a \psi) &:= \mathrm{sd}(\psi), & \mathrm{sd}(\Diamond\!\!\!\!-_a \psi) &:= \mathrm{sd}(\psi) + 1.
\end{aligned}
$$

In the next section we will see that the sabotage depth of a formula is the main factor in the complexity of the model checking problem for SML.

3 Model Checking for SML

In this section we consider the model checking problem for SML. The general question in model checking is whether a given structure is a model of a given formula. The model checking problem for modal logic (ML) over transition systems is known to be solvable in polynomial time (cf. [2]).

Proposition 1. *The model checking problem for* ML *is* PTIME*-complete and can be solved in time* $\mathcal{O}(|\varphi| \cdot |\mathcal{T}|)$*, where* $|\varphi|$ *is the size of the given* ML*-formula* φ *and* $|\mathcal{T}|$ *is the size of the given transition system* $\mathcal{T}$*.* □

The combined complexity of model checking for SML, i.e., the complexity measured in terms of the size of the formula and in the size of the structure, was already settled in [5].

Theorem 2. *Model checking for* SML *is* PSPACE*-complete.* □

In many cases one of the inputs for the model checking problem is fixed. If one wants to verify a single property for several systems the formula is fixed and if there are different properties that have to be verified for a single system the structure is fixed. For modal and temporal logics these two views of the complexity are usually referred to as program complexity and formula complexity of the model checking problem.

In the following we show that the model checking problem for SML with one of the inputs fixed (either the transition system or the formula) can be solved in linear, resp., in polynomial time. For this purpose we reduce the model checking problem for SML to the model checking problem for ML and show that this reduction can be done in linear, resp., in polynomial time if either the transition system or the formula is fixed.

Let $\mathcal{T} = (S, \Sigma, R, L)$ be a transition system. We define a new transition system $\mathcal{T}^{\mathrm{sab}} = (S^{\mathrm{sab}}, \Sigma^{\mathrm{sab}}, R^{\mathrm{sab}}, L^{\mathrm{sab}})$ that encodes all possible ways of sabotaging $\mathcal{T}$:

$$
\Sigma^{\mathrm{sab}} := \Sigma \,\dot\cup\, \{\bar{a} \mid a \in \Sigma\}, \quad S^{\mathrm{sab}} := S \times 2^R,
$$

$$R^{\mathrm{sab}} := \{((s_1, E), a, (s_2, E)) \mid (s_1, a, s_2) \in R \setminus E\} \cup \{((s, E_1), \bar{a}, (s, E_2)) \mid \exists s_1, s_2 \in S\ (E_2 = E_1 \dot\cup \{(s_1, a, s_2)\})\},$$

$$L^{\mathrm{sab}}(s, E) := L(s) \text{ for each } s \in S \text{ and } E \subseteq R.$$

Over this system one can express the sabotage operator $\Diamond\!\!\!\!-_a$ by traversing an $\bar{a}$ edge, i.e., by the modal operator $\Diamond_{\bar{a}}$. This motivates the following inductive definition of the ML-formula $\widehat{\varphi}$ for a given SML-formula φ:

$$\widehat{\varphi} = \begin{cases} \varphi & \text{if } \varphi = \top \text{ or } \varphi = p, \\ \widehat{\varphi_1} \vee \widehat{\varphi_2} & \text{if } \varphi = \varphi_1 \vee \varphi_2, \\ \neg\widehat{\psi} & \text{if } \varphi = \neg\psi, \\ \Diamond_a \widehat{\psi} & \text{if } \varphi = \Diamond_a \psi, \\ \Diamond_{\bar{a}} \widehat{\psi} & \text{if } \varphi = \Diamond\!\!\!\!-_a \psi. \end{cases}$$

If the sabotage depth of a formula φ is small then we do not need the complete transition system $\mathcal{T}^{\mathrm{sab}}$ to evaluate $\widehat{\varphi}$. So, for $n \in \mathbb{N}$, we define $\mathcal{T}^{\mathrm{sab}}_n$ to be the transition system $\mathcal{T}^{\mathrm{sab}}$ restricted to the states (s, E) with $|E| \leq n$. Note that $\mathcal{T}^{\mathrm{sab}}_0$ is isomorphic to $\mathcal{T}$ and $\mathcal{T}^{\mathrm{sab}}_n = \mathcal{T}^{\mathrm{sab}}$ for $n \geq |R|$.

Lemma 3. *Let $\mathcal{T} = (S, \Sigma, R, L)$ be a transition system and φ be an* SML*-formula. Then $(\mathcal{T}, s) \models \varphi$ iff $(\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\varphi)}, (s, \emptyset)) \models \widehat{\varphi}$.*

Proof. We show by induction on the structure of φ that for each $E \subseteq R$:

$$(\mathcal{T}_E, s) \models \varphi \Leftrightarrow (\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\varphi)+|E|}, (s, E)) \models \widehat{\varphi}.$$

For $E = \emptyset$ we obtain the desired property. The only interesting case for this induction is for $\varphi = \Diamond\!\!\!\!-_a \psi$. The definitions of the semantics of the sabotage operator and the structure $\mathcal{T}_E$ imply that $(\mathcal{T}_E, s) \models \Diamond\!\!\!\!-_a \psi$ iff there exists an edge $(s_1, a, s_2) \in R \setminus E$ such that $(\mathcal{T}_{E'}, s) \models \psi$ for $E' = E \dot\cup \{(s_1, a, s_2)\}$. By the induction hypothesis this holds iff $(\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\psi)+|E'|}, (s, E')) \models \widehat{\psi}$. Since $\mathrm{sd}(\psi)+|E'| = (\mathrm{sd}(\varphi) - 1) + (|E| + 1) = \mathrm{sd}(\varphi) + |E|$ and since there is an $\bar{a}$-edge from (s, E) to (s, E') we get $(\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\psi)+|E'|}, (s, E')) \models \widehat{\psi}$ iff $(\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\varphi)+|E|}, (s, E)) \models \Diamond_{\bar{a}} \widehat{\psi}$. This implies the claim because $\widehat{\varphi} = \Diamond_{\bar{a}} \widehat{\psi}$. □

This reduction can be used to determine the formula complexity and the program complexity of SML model checking.

Theorem 4. *The model checking problem for* SML *with a fixed transition system can be solved in linear time in the size of the formula (formula complexity). The model checking problem for a fixed* SML*-formula can be solved in polynomial time in the size of the transition system (program complexity).*

Proof. By Proposition 1 and Lemma 3 we can solve the model checking problem for φ and $\mathcal{T}$ in time $\mathcal{O}(|\widehat{\varphi}| \cdot |\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\varphi)}|)$. From the definition of $\widehat{\varphi}$ we get $|\widehat{\varphi}| = |\varphi|$. For a fixed transition system $\mathcal{T}$ we can estimate the size of $\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\varphi)}$ by $|\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\varphi)}| \in \mathcal{O}(|\mathcal{T}| \cdot 2^{|\mathcal{T}|})$. Hence the formula complexity is in $\mathcal{O}(|\varphi|)$.

Since the number of subsets $E \subseteq R$ with $|E| \leq \mathrm{sd}(\varphi)$ is in $\mathcal{O}(|\mathcal{T}|^{\mathrm{sd}(\varphi)})$ we obtain for a fixed SML-formula $|\mathcal{T}^{\mathrm{sab}}_{\mathrm{sd}(\varphi)}| \in \mathcal{O}(|\mathcal{T}|^{\mathrm{sd}(\varphi)+1})$. So the program complexity is polynomial in $|\mathcal{T}|$. □

4 Model-Theoretic Properties of SML

For many logics, e.g. temporal logics like CTL, CTL*, and LTL, satisfiability can be shown to be decidable using the small model property. A logic has the small model property if a formula from this logic is satisfiable iff it has a model of size bounded by a computable function of the size of the formula. To decide the satisfiability problem, provided that the model checking problem is decidable, it is sufficient to check all structures up to this bounded sized. Modal logic even has the small tree model property, i.e., for each satisfiable ML-formula there exists a small tree that is a model of the formula. In this section we analyse model-theoretic properties of this kind for SML.

In the sequel let $\mathcal{T} = (S, \Sigma, R, L)$ be a transition system and $s \in S$. For any subset $A \subseteq \Sigma$ we fix the SML-formula

$$\sigma_A :\equiv \Diamond_A \top \equiv \bigvee_{a \in A} \Diamond_a \top$$

expressing that there is an a-successor of the current state for some $a \in A$. For a single letter we write σ_a instead of $\sigma_{\{a\}}$. For a word $\alpha = a_1 \dots a_k$ we set $\sigma_\alpha :\equiv \Diamond_{a_1} \dots \Diamond_{a_k} \top$. Since we deal with pointed transition systems $(\mathcal{T}, s)$ we use the notation 'a-successor' without a reference to a state as an abbreviation for 'a-successor of the origin s'. For later use we define the following SML-formulae. For $a \in \Sigma$ and $n \in \mathbb{N}$ let

$$\gamma_{0,a} :\equiv \neg\sigma_a, \quad \gamma_{1,a} :\equiv \sigma_a \wedge \Diamond\!\!\!\!-_a \neg\sigma_a, \quad \gamma_{n+2,a} :\equiv \boxminus_a^{n+1} \sigma_a \wedge \Diamond\!\!\!\!-_a^{n+2} \neg\sigma_a$$

We write γ_a as an abbreviation for $\gamma_{1,a}$. It is easy to see that we can fix the number of a-successors for a given state by the SML-formula $\gamma_{n,a}$:

Lemma 5. $(\mathcal{T}, s) \models \gamma_{n,a}$ *iff state* s *has exactly* n *different* a*-successors.* □

In contrast to modal logic, SML lacks the tree model property, i.e., there are satisfiable SML-formulae which do not have a tree model.

Lemma 6. *The logic* SML *does not have the acyclic model property. In particular it does not have the tree model property and it is not bisimulation-invariant.*

Proof. Consider the SML-formula $\varphi :\equiv \sigma_{aa} \wedge \boxminus_a \neg\sigma_a$. Then every transition system $\mathcal{T}$ with $(\mathcal{T}, s) \models \varphi$ has only one a-transition which starts and ends in state s. The last property holds since every bisimulation-invariant logic over transition systems has the tree model property. □

Another difference to modal logic is that each satisfiable ML-formula has a finite model, whereas for SML this property does not hold.

Theorem 7. *The logic* SML *does not have the finite model property.*

Proof. Let φ be the following SML-formula:

$$\Diamond_g \neg\sigma_a \wedge \Diamond\!\!\!-_g \Box_g \sigma_a \wedge \tag{M1}$$
$$\Box_g \Diamond\!\!\!-_a \neg\sigma_a \wedge \tag{M2}$$
$$\Box_g \gamma_h \wedge \tag{M3}$$
$$\Box_g \Box_a \gamma_h \wedge \tag{M4}$$
$$\neg\sigma_{\{a,h\}} \wedge \Box_g \neg\sigma_g \wedge \tag{M5}$$
$$\boxminus_h \Diamond_g (\sigma_h \wedge \Diamond_a \neg\sigma_h) \tag{M6}$$

Then every transition system $\mathcal{T}$ with $(\mathcal{T}, s) \models \varphi$ has the following properties. By M1: there is exactly one g-successor which has no a-successors and all other g-successors have an a-successor. In particular there is a g-successor. By M2: each g-successor has at most one a-successor. By M3: each g-successor has exactly one h-successor. In particular there is an h-transition. By M4: each g-a-successor has exactly one h-successor. By M5: the origin has no a- or h-successors and there are no g-g-successors. Finally M6 expresses that for every deleted h-transition there is a g-successor in the corresponding submodel which has an h-successor v and an a-successor w such that w has no h-successors (maybe with $v = w$).

It is easy to see that the transition system depicted in Fig. 1 is an infinite model of φ (if pointed at state s). For the last property notice that, if the nth h-transition from the left is deleted, then the $n+1$th g-transition from the left satisfies M6.

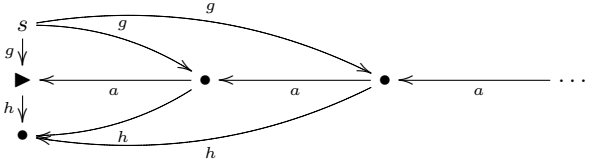

Fig. 1. An infinite model of φ

In the sequel the unique g-successor without a-successors is called the *sink state* and is displayed as ▶. Further we omit all g- and h-transitions. This means that in the figures all displayed vertices are g-successors from the origin and have h-transitions leading to a separate vertex.

We have to show that φ has only infinite models. For that we claim:

Claim 1. For every model $\mathcal{T}$ with $(\mathcal{T}, s) \models \varphi$ we have that every h-transition starts in a g-a-successor of s.

Proof (of Claim 1). Assume that there is an h-transition which starts in a state which is not a g-a-successor of s. After deleting this h-transition, M4 is still valid, i.e., there is no g-a-successor of s without an h-successor contradicting the second part of M6.

Claim 2. For every model $\mathcal{T}$ with $(\mathcal{T}, s) \models \varphi$ the state s has infinitely many g-successors.

Proof (of Claim 2). Assume that s has only k many g-successors for some $k \in \mathbb{N}$. Because of M1 and M2 state s has at most $k-1$ many g-a-successors. Hence there exists a g-successor v of s which is not a g-a-successor of s. Due to Property M3 state v has an h-successor, but then – by Claim 1 – state v has to be a g-a-successors of s, contradiction.

This completes the proof of the theorem. □

Note that not all given conjuncts are necessary to obtain an infinite model, but we need this construction below. Further the middle part of a model does not need to be a single chain which is only unbounded to the right. For example we could have further chains which are unbounded to both sides. Other models are 'inverse infinite trees'.

5 Undecidability of Satisfiability for SML

In this section we show that the satisfiability problem for SML is undecidable. To be more precise, we show that the problems of deciding whether a given formula 1. has a model (*Satisfiability*), 2. has a finite model (*Finite Satisfiability*), and 3. is satisfiable, but has only infinite models (*Infinity Axiom*) are undecidable. To that aim we first define three variants of *Post's Correspondence Problem* (cf. [4]) that will be reduced to the mentioned problems.

We fix an alphabet Σ with $|\Sigma| \geq 2$. Given two lists $\bar{\alpha} = (\alpha_1, \dots, \alpha_n)$ and $\bar{\beta} = (\beta_1, \dots, \beta_n)$ of non-empty words over Σ with $n \geq 2$ we formulate the following correspondence problems:

1. $(\bar{\alpha}, \bar{\beta}) \in \mathrm{PCP}_*$ iff there is a finite sequence $(i_1, \dots, i_k)$ in $\{2, \dots, n\}$ such that the finite words $\alpha_1\alpha_{i_1} \dots \alpha_{i_k}$ and $\beta_1\beta_{i_1} \dots \beta_{i_k}$ are the same,
2. $(\bar{\alpha}, \bar{\beta}) \in \mathrm{PCP}_\omega$ iff $(\bar{\alpha}, \bar{\beta}) \notin \mathrm{PCP}_*$ and there is a infinite sequence $(i_1, i_2, \dots)$ in $\{2, \dots, n\}$ such that the ω-words $\alpha_1\alpha_{i_1}\alpha_{i_2} \dots$ and $\beta_1\beta_{i_1}\beta_{i_2} \dots$ are the same,
3. $\mathrm{PCP}_\infty := \mathrm{PCP}_* \cup \mathrm{PCP}_\omega$.

We require that both decompositions start with α_1, resp., β_1 and that the index 1 does not occur again. The sequence $(i_1, \dots, i_k)$, resp., $(i_1, i_2, \dots)$ is called then a finite, resp., an infinite solution. Note that there are two different sorts of infinite solutions: regular solutions where the sequence is ultimately periodic and irregular solutions. The usual proof of undecidability of (modified) PCP can be easily adapted to show the undecidability of these three correspondence problems (cf. [4]).

Given an instance $(\bar{\alpha}, \bar{\beta})$ of the correspondence problems let $\Gamma := \Sigma \cup \{\#\}$, $I := \{1, \dots, n\}$ and $\Delta := \Gamma \cup I \cup \{g, h\}$ (w.l.o.g. $\Sigma \cap I = \emptyset$ and $\Sigma \cap \{g, h, \#\} = \emptyset$). To show the undecidability of the satisfiability problems we code solutions of the correspondence problems within a transition system over the alphabet Δ which

is SML-definable. The models are similar to the ones of Theorem 7, but in contrast to them they may be finite: we can interrupt the building process by an 'end marker' #. The idea is to represent a solution on the middle chain of the models, reading it from right to left starting at the marker #. The words of the first list $\bar{\alpha}$ start and end in vertices with an I-successor, the corresponding words of the second list $\bar{\beta}$ start and end in vertices, which are appropriate I-successors. Figure 2 shows as an example the representation of the finite solution $(1, 3, 2)$ for the lists $\bar{\alpha} = (ab, b, abaab)$ and $\bar{\beta} = (aba, abb, ba)$. Note that all words are read from right to left. If the solution is finite both decompositions end in the unique ▶-vertex and the complete model can be made finite. If the solution is infinite the decomposition never ends and we have an infinite model.

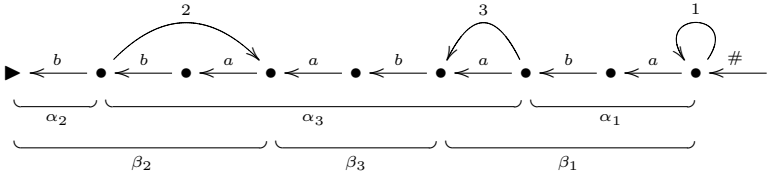

Fig. 2. Representation of a finite solution

Before we give the SML-specifications to realise this idea we introduce some auxiliary notations. For non-empty words α over Σ and an SML-formula ψ we define inductively $\delta_\alpha(\psi)$ by $\delta_a(\psi) := \Diamond_a\psi$ and $\delta_{a\alpha'}(\psi) := \Diamond_a(\neg\sigma_I \wedge \delta_{\alpha'}(\psi))$ ($a \in \Sigma$, $\alpha' \in \Sigma^+$). For example $\delta_{abb}(\top)$ expresses that there is an a-b-b-successor, but neither an a-I- nor an a-b-I-successor. For a non-empty word $\beta = b_1 \dots b_k$ over Σ and an SML-formula ψ let $\delta'_\beta(\psi) := \Diamond_{b_1} \dots \Diamond_{b_k}\psi$.

To code solutions of the correspondence problems let $\varphi_{\bar{\alpha},\bar{\beta}}$ be the following SML-formula (we explain the several subformulae below):

$$\Diamond_g\neg\sigma_\Gamma \wedge \hat{\Diamond}_g\Box_g\sigma_\Gamma \wedge \tag{S1}$$

$$\Box_g\hat{\Diamond}_\Gamma\neg\sigma_\Gamma \wedge \Box_g\hat{\Diamond}_I\neg\sigma_I \wedge \tag{S2}$$

$$\Box_g\gamma_h \wedge \Box_g\Box_\Gamma\gamma_h \wedge \boxminus_h\Diamond_g\neg\sigma_h \wedge \tag{S3}$$

$$\neg\sigma_{\Delta\setminus\{g\}} \wedge \Box_g\neg\sigma_g \wedge \tag{S4}$$

$$\boxminus_h\Diamond_g\big((\sigma_\# \wedge \neg\sigma_h) \vee (\sigma_h \wedge \Diamond_\Gamma\neg\sigma_h)\big) \wedge \tag{S5}$$

$$\sigma_{g\#1} \wedge \boxminus_\#\Box_g\neg\sigma_\# \wedge \neg\sigma_{g\Gamma\#} \wedge \tag{S6}$$

$$\Diamond_g(\sigma_{11} \wedge \boxminus_1\neg\sigma_1) \wedge \tag{S7}$$

$$\Box_g\Big[\bigwedge_{i\in I}\Big[\sigma_i \to \Big[\big(\delta_{\alpha_i}(\neg\sigma_\Gamma) \wedge \Diamond_i\delta'_{\beta_i}(\neg\sigma_\Gamma)\big) \vee \bigvee_{j\in I, j\neq 1}\big(\delta_{\alpha_i}(\sigma_{jh}) \wedge \Diamond_i\delta'_{\beta_i}(\sigma_h) \wedge \hat{\Diamond}_h\big(\delta_{\alpha_i}(\Diamond_j\neg\sigma_h) \wedge \Diamond_i\delta'_{\beta_i}(\neg\sigma_h)\big)\big)\Big]\Big]\Big]. \tag{S8}$$

S1 expresses that there is exactly one g-successor which has no Γ-successors and all other g-successors have a Γ-successor, whereas S2 ensures that each g-

successor has at most one Γ-successor and at most one I-successor. S3 requires that each g-successor and each g-Γ-successor has exactly one h-successor and every h-transition starts at a g-successor. S4 expresses that the origin has only g-successors and that there are no g-g-successors. If S5 is true then for every deleted h-transition there is a g-successor in the corresponding submodel which either has a #-successor, but no h-successors; or has an h-successor and a Γ-successor v such that v has no h-successors. The next formula S6 ensures that each model has a g-#-successor w, but only one #-transition at all and that there is no g-Γ-#-successor. Further S6 and S7 require that there is exactly one 1-transition which starts and ends in w.

Notice that we do not need to have a finite Σ-chain starting from the (unique) g-#-successor. Finally, S8 expresses (together with the previous formulae) that for every g-successor v, whenever v has an i-successor for some $i \in I$ then either one reaches the unique sink state of the model by both words α_i and $i \cdot \beta_i$ (see Fig. 3) or there is some $j \in I$, $j \neq 1$ such that one reaches the *same* g-successor by both words $\alpha_i \cdot j$ and $i \cdot \beta_i$ (see Fig. 4). Further, while reading the word α_i, there are no I-successors except for the first and the last vertex.

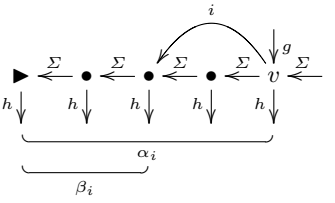

Fig. 3. First case for S8

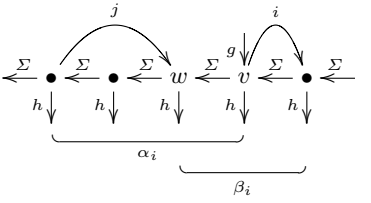

Fig. 4. Second case for S8

In the sequel we show some properties which are valid for every transition system $\mathcal{T}$ with state $s \in S$ such that $(\mathcal{T}, s) \models \varphi_{\bar{\alpha},\bar{\beta}}$.

Property 1. (1) Every g-successor with an I-successor has a Σ-successor as well. (2) Each g-Γ-successor is also a g-successor. (3) There is a unique g-#-successor v. (4) The unique sink state has no I-successors. In particular the g-#-successor is distinct from the sink state.

Proof. (1) If g-successor v has an i-successor then, due to S8, $\delta_{\alpha_i}(\top)$ is true. In particular $\Diamond_{a_1^i} \top$ holds, if $\alpha_i = a_1^i \ldots$ (α_i is non-empty). Hence v has an a_1^i-successor. (2) By S3 every g-Γ-successor has an h-successor, but every h-transition starts at a g-successor. (3) S6 ensures the existence and, since there is only one #-transition at all, the uniqueness of v. (4) Immediately by (1) and since the g-#-successor has a 1-successor. □

Now we inductively define a (finite or infinite) sequence v_k of vertices which can be found in every model of $\varphi_{\bar{\alpha},\bar{\beta}}$. Let v_0 be the unique g-successor that has a #-successor and let v_1 be this #-successor (well-defined by Prop. 1.3). By Prop. 1.2,

v_1 is also a g-successor. Further we have $v_0 \neq v_1$ (otherwise $\sigma_{g\Gamma\#}$ holds which violates S6). Assume that v_k for $k \geq 1$ is already defined such that v_k is the unique Γ-successor of v_{k-1}. In particular v_k is a g-successor by Prop. 1.2. *Case 1*: v_k has no Σ-successor. Due to $\neg\sigma_{g\Gamma\#}$, v_k has no #-successor as well. Since the sink state is the unique g-successor without a Γ-successor we have that v_k is the sink state. In this case let $\kappa := k+1$. *Case 2*: v_k has a Σ-successor. Let v_{k+1} be this Σ-successor which is unique by S2.

If v_k is defined for all $k \in \omega$ then set $\kappa := \omega$. Note that the v_k's do not need to be pairwise distinct, since we may have a loop in the Σ-chain. Now we extract those v_k's which have I-successors. For that we inductively define the sequence j_k. Let $j_0 := 1$. Assume that j_k is already defined such that v_{j_k} has an I-successor. If there is m with $j_k < m < \kappa$ such that v_m has an I-successor then let j_{k+1} be the minimal m with this property. Otherwise let $\lambda := k+1$.

Again, if j_k is defined for all $k \in \omega$ then set $\lambda := \omega$. For all $k < \lambda$ we have: the I-successor of v_{j_k} is unique by S2. Hence we set $i_k = i$ if v_{j_k} has an i-successor.

Property 2. $\kappa < \omega$ *iff* $\lambda < \omega$.

Proof. ($\Rightarrow$) This is clear by the definition of the j_k's. ($\Leftarrow$) If $\lambda < \omega$ let $m = \lambda - 1$ (note that $\lambda \geq 1$ by definition). Then v_{j_m} is the last vertex in the sequence $v_1, v_2, \ldots$ which has an I-successor. But for v_{j_m} the last disjuncts of S8 cannot be satisfied, since otherwise – after $|\alpha_{i_m}|$ steps – we would reach $v_{j_m+|\alpha_{i_m}|}$ which must have an j-successor for some $j \in I, j \neq 1$. So only the first disjunct is satisfied and $v_{j_m+|\alpha_{i_m}|}$ is the sink state, i.e., $\kappa = j_m + |\alpha_{i_m}| + 1$. □

If $\lambda < \omega$ we set $j_\lambda := \kappa - 1$, i.e., in this case v_{j_λ} is the sink state.

Property 3. (1) $j_0 = 1$ *and* $j_k \in \{2, \ldots, n\}$ *for all* $k \geq 1$. *For all* $k < \lambda$ *the following holds: (2)* $j_k < j_{k+1}$, *(3)* v_{j_k} *has* α_{i_k}*-successor* $v_{j_{k+1}}$, *and (4) for all* $j_k < m < j_{k+1}$, v_m *has no* I*-successors. (5) The* i_k*-successor of* v_{j_k} *is equal to* v_m *for some* $1 \leq m < \kappa$.

Proof. (1) By the definition of the j_k's and by S8. (2) is clear by the definition of the j_k's. (3) and (4) immediately follow from S8 and the definition of δ_α. We show (5) by induction. We have $j_0 = 1$ and $i_0 = 1$. S6 and S7 ensure that the 1-successor of v_1 is v_1 itself. Assume now that the property is already given for k and that $k+1 < \lambda$. S8 guarantees that we reach from vertex v_{j_k} the *same* vertex by the words $\alpha_{i_k} \cdot i_{k+1}$ and $i_k \cdot \beta_{i_k}$. Since by (3), the α_{i_k}-successor of v_{j_k} is exactly $v_{j_{k+1}}$ this means that the i_{k+1}-successor of $v_{j_{k+1}}$ is equal to the $i_k \cdot \beta_{i_k}$-successor of v_{j_k}. By induction the i_k-successor of v_{j_k} is equal to v_m for some $1 \leq m < \kappa$, hence the i_{k+1}-successor of $v_{j_{k+1}}$ is equal to $v_{m'}$ with $m' = m + |\beta_{i_k}| < \kappa$. □

The last property ensures that we stay on the Σ-chain starting at v_1 if we follow the I-transitions and that we do not change to other parts, resp., branches within the model. Now we can define the sequence l_k for $k < \lambda$ by $l_k = m$, if v_m is the (unique) i_k-successor of v_{j_k}. Again, if $\lambda < \omega$ we set $l_\lambda := \kappa - 1$. Then we have:

Property 4. (1) $l_0 = 1$. *For all* $k < \lambda$ *the following holds: (2)* $l_k < l_{k+1}$, *and (3)* v_{l_k} *has* β_{i_k}*-successor* $v_{l_{k+1}}$.

Proof. The 1-successor of v_1 is v_1, hence (1) holds. By Prop. 3.3 and the definition of l_k the i_k-successor of v_{j_k} is v_{l_k} and the $\alpha_{i_k} \cdot i_{k+1}$-successor of v_{j_k} is $v_{l_{k+1}}$. On the other hand, by S8, the $i_k \cdot \beta_{i_k}$-successor of v_{j_k} is $v_{l_{k+1}}$ as well. This shows (3) and, since β_{i_k} is non-empty, also (2). □

Finally, we state the main property (we denote the prefix of a word α of length k by $\alpha[k]$):

Property 5. (1) If $\kappa < \omega$ then $\alpha_1\alpha_{i_1} \dots \alpha_{i_{\lambda-1}} = \beta_1\beta_{i_1} \dots \beta_{i_{\lambda-1}}$. (2) If $\kappa = \omega$ then $(\alpha_1\alpha_{i_1}\alpha_{i_2} \dots)[m] = (\beta_1\beta_{i_1}\beta_{i_2} \dots)[m]$ for all $m \in \omega$.

Proof. (1) By Prop. 2 we also have $\lambda < \omega$, in particular j_λ and l_λ are defined. Since $v_{j_0} = v_1$ and v_{j_λ} is the sink state, by Prop. 3.3 we have that $\alpha_1\alpha_{i_1} \dots \alpha_{i_{\lambda-1}}$ is exactly the Σ-word between the vertices v_1 and the sink state. On the other hand, since $v_{l_0} = v_1$ and v_{l_λ} is the sink state, by Prop. 4.3 we have that $\beta_1\beta_{i_1} \dots \beta_{i_{\lambda-1}}$ also is the Σ-word between the vertices v_1 and the sink state. In particular the two words are identical.

(2) For $m = 0$ this is clear. For $m > 0$ let $\mu_m := \max\{k \mid j_k \leq m\}$ and $\nu_m := \max\{k \mid l_k \leq m\}$. Again by Prop. 3.3, the word between the vertices v_1 and v_{m+1} is exactly the Σ-word $\alpha_1 \dots \alpha_{i_{\mu_m-1}} \cdot (\alpha_{i_{\mu_m}}[m + 1 - j_{\mu_m}]) = (\alpha_1\alpha_{i_1}\alpha_{i_2} \dots)[m]$. On the other hand, again by Prop. 4.3, this word also is $\beta_1 \dots \beta_{i_{\nu_m-1}} \cdot (\beta_{i_{\nu_m}}[m + 1 - l_{\nu_m}]) = (\beta_1\beta_{i_1}\beta_{i_2} \dots)[m]$. □

Now we are ready to show the main theorem:

Theorem 8. *The following holds:*

1. $(\bar{\alpha}, \bar{\beta}) \in \mathrm{PCP}_*$ *iff* $\varphi_{\bar{\alpha},\bar{\beta}}$ *has a finite model,*
2. $(\bar{\alpha}, \bar{\beta}) \in \mathrm{PCP}_\omega$ *iff* $\varphi_{\bar{\alpha},\bar{\beta}}$ *has an infinite, but no finite model,*
3. $(\bar{\alpha}, \bar{\beta}) \in \mathrm{PCP}_\infty$ *iff* $\varphi_{\bar{\alpha},\bar{\beta}}$ *is satisfiable.*

Fig. 5. An infinite model

Proof. (1) (⇒) Let $(i_1, \dots, i_k)$ be a finite solution. Then the appropriate finite chain analogous to the one depicted in Fig. 2 together with the corresponding g- and h-transition as depicted in Fig. 1 is a finite model of $\varphi_{\bar{\alpha},\bar{\beta}}$ (if pointed at vertex s). (⇐) If $\varphi_{\bar{\alpha},\bar{\beta}}$ has a finite model then for that model we have $\kappa < \omega$. By Prop. 3.1 and Prop. 5.1 this means that $(i_1, \dots, i_{\lambda-1})$ is a finite solution and therefore $(\bar{\alpha}, \bar{\beta}) \in \mathrm{PCP}_*$.

(2) (⇒) Let $(i_1, i_2, \dots)$ be an infinite solution and let $a \in \Sigma$ be arbitrary. Then the transition system depicted in Fig. 5 is a model of $\varphi_{\bar{\alpha},\bar{\beta}}$ (together with the corresponding g- and h-transitions). The upper infinite chain is labelled with a's and the vertices have no I-successors. The lower infinite chain is labelled with

the word $\alpha_1\alpha_{i_1}\alpha_{i_2}\dots$ (from right to left starting from the g-#-successor) and has appropriate I-transitions as described above.

By definition of PCP_ω we have $(\bar{\alpha},\bar{\beta}) \notin \mathrm{PCP}_*$ and therefore by (1), $\varphi_{\bar{\alpha},\bar{\beta}}$ has no finite model. ($\Leftarrow$) If $\varphi_{\bar{\alpha},\bar{\beta}}$ has a infinite model then for that model we have $\kappa = \omega$. Since there are no finite models we have $(\bar{\alpha},\bar{\beta}) \notin \mathrm{PCP}_*$ by (1). By Prop. 3.1 and Prop. 5.2 this means that $(i_1, i_2, \dots)$ is an infinite solution and therefore $(\bar{\alpha},\bar{\beta}) \in \mathrm{PCP}_\omega$.

(3) This is immediate by (1) and (2). □

Applying the undecidability of the correspondence problems we immediately obtain the desired result:

Theorem 9. *The problems* Satisfiability*,* Finite Satisfiability*, and* Infinity Axiom *for* SML *are undecidable.* □

6 Conclusion and Outlook

We have considered the sabotage modal logic SML, an extension of modal logic that is capable of describing elementary changes of structures. Modal logic itself is one of the simplest logics for specifying properties of transition systems. We have shown that operators that capture basic changes of the structure, namely the removal of edges, already strengthen modal logic in such a way that all the nice algorithmic and model-theoretic properties of modal logic get lost. In fact, from the viewpoint of complexity and model theory SML much more resembles first-order logic than modal logic, except for the linear time formula complexity of model checking.

There are some open questions related to SML. For example one may restrict the global power of the sabotage operator (e.g., the deleted transition has to start at the current state). Model checking of SML for unary alphabets is still PSPACE-complete (cf. [5]), but it is open whether satisfiability is then decidable. Also, it would be nice to have a characteristic notion of bisimulation for SML and to know more about valid principles, e.g. interaction laws between ordinary and sabotage modalities.

References

1. J. v. Benthem. An essay on sabotage and obstruction. In D. Hutter and S. Werner, editors, *Festschrift in Honour of Prof. Jörg Siekmann*, LNAI. Springer, 2002.
2. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.
3. E. Grädel. Finite model theory and descriptive complexity. In *Finite Model Theory and Its Applications.* Springer, 2003. To appear.
4. J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 2001.
5. C. Löding and P. Rohde. Solving the sabotage game is PSPACE-hard. In *Proceedings of MFCS 2003.* Springer, 2003.
6. P. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic – Proceedings of AiML'2002.* World Scientific, 2003. To appear.