The Modal Logics of the Poison Game

Francesca Zaffora Blando[*] Krzysztof Mierzewski[*] Carlos Areces[†]

Abstract

The poison game is a two-player zero-sum game played on directed graphs, first introduced by Duchet and Meyniel [1993] in the context of graph theory, where one of the players travels along the edges of a graph, while the other modifies the underlying structure by marking vertices. In this paper, we investigate the poison game from the perspective of modal logic, as a natural case study of the use of modal languages equipped with model-changing operators to describe evolving relational structures. In particular, to model the poison game, we consider three memory logics of decreasing expressive power but increasing fit with the game. We begin with $\mathcal{ML}_\emptyset$, the basic memory logic restricted to the initial class of models with an empty memory (see [Areces et al., 2011]). We then identify two fragments of $\mathcal{ML}_\emptyset$, which we respectively denote as $\mathcal{PML}$ and $\mathcal{PSL}$, and whose modal operators capture operations on models that mimic more closely the moves of both players. We show that these logics form a chain in expressive power with $\mathcal{PSL} < \mathcal{PML} < \mathcal{ML}_\emptyset$, and we introduce suitable notions of bisimulation for the two new logics presented in this paper. We then show that model checking for both $\mathcal{PML}$ and $\mathcal{PSL}$ is PSPACE-complete. The construction also establishes that determining the existence of a winning strategy in the poison game is PSPACE-hard. We conclude by proving that $\mathcal{PML}$, while strictly less expressive than $\mathcal{ML}_\emptyset$, nonetheless has an undecidable satisfiability problem.

1 Introduction

Logic and games have gone hand in hand for quite some time. As noted by Hodges [2013], one can already find connections between logic and argumentation games in Aristotle's work on syllogisms. Nowadays, logical games are used in a multitude of settings. There are games for model comparison [Fraïssé, 1954; Ehrenfeucht, 1961],

[*]Department of Philosophy, Stanford University and Logical Dynamics Lab, Center for the Study of Language and Information, Stanford, California, USA.

[†]Facultad de Matemática, Astronomía, Física y Computación, Universidad Nacional de Córdoba and CONICET, Córdoba, Argentina.

argumentation and dialogue [Lorenzen, 1955], model checking [Hintikka, 1973], as well as for building models for a given formula [Hodges, 2006].

A more recent strand of research tackles the opposite direction: in addition to using game-theoretic tools in logic, one can also focus on the use of logical languages for analysing games (see [van Benthem, 2014]). For instance, games of imperfect information may be naturally modelled within epistemic or doxastic logic, and certain common computational tasks might be 'gamifiable', which then facilitates their analysis from the perspective of modal logic. An example of this latter application is the *sabotage game* introduced by van Benthem [2005], a two-player zero-sum game played on graphs. In a sabotage game, one player tries to get from one vertex to another fixed set of vertices, while the other player tries to prevent this by deleting edges in the graph. In other words, a sabotage game is a version of the *graph reachability* problem involving an edge-deleting (or sabotaging) player, whose goal is rendering the target vertices inaccessible.

In order to model this game, van Benthem [2005] introduces a modal calculus, *sabotage modal logic*, which differs from the basic modal language, in that it is equipped with a transition-deleting modality which modifies the underlying model. This means that sabotage modal logic, as opposed to standard modal logic, can express changes of transition systems, on top of the usual properties of static models.[1]

In this paper, we consider another two-player zero-sum game played on graphs called the *poison game*—first introduced by Duchet and Meyniel [1993]—from a modal perspective. While the sabotage game is a logic-inspired graph game, the poison game comes from graph theory: studying the poison game is therefore a good way of testing whether methods from modal logic can be fruitfully extended to a wider domain.

Let (G, R) be a directed graph (with no double edges) and $s \in G$ a distinguished *starting vertex*. The poison game proceeds as follows: the two players—Traveller and Poisoner—alternate their moves and, at each step, choose a vertex that is a successor of the vertex previously chosen by the other player. The game begins at the starting vertex s.[2] Poisoner makes the first move by choosing a successor s' of s (that is, a point $s' \in G$ such that $(s, s') \in R$), which she then poisons with a poison that affects exclusively Traveller. This means that Poisoner's move renders vertex s' inaccessible to Traveller, but not to Poisoner. Then, Traveller has to choose a non-poisoned successor of s', and so on. The winning conditions are as follows. Traveller wins the poison game if either (i) she manages to keep choosing non-poisoned successors no matter what vertices Poisoner selects, or (ii) she begins her turn at a vertex with no successors, or

[1]For a detailed discussion of sabotage modal logic and other relation-changing logics, see, for instance, [Löding and Rohde, 2003a,b; Aucher et al., 2015, 2017; Areces et al., 2015].

[2]Here we treat the starting position as being given. In the original formulation of the game [Duchet and Meyniel, 1993], Traveller makes the first move by *choosing* a starting vertex $s \in G$.

(iii) Poisoner begins her turn at a vertex with no successors. If at least one of (i)-(iii) obtains, Traveller *survives* the game. Poisoner, on the other hand, wins the poison game if she manages to poison a vertex that (a) has at least one successor and (b) all of whose successors have already been poisoned (we then say that Traveller gets poisoned, as she would be forced to move to a poisoned vertex in the next round).

The landscape of modal logics is rich and varied. When designing a modal system to model a graph game like the poison game, one is immediately confronted with the question of which language is best suited for capturing the game, and of which minimal requirements a logic should meet in order to qualify as a plausible contender. For instance, it should be possible, within the logic, to talk about the moves of both players and to express, at least approximately, the existence of winning strategies.

In this modal setting, where we evaluate formulas at individual vertices in the graph, we are particularly interested in describing games from a local perspective. In the poison game, the players construct a path through the graph, moving from vertex to vertex: modal languages are singularly well-suited to describe the current stage of the game, step-by-step, by capturing local (and possibly dynamic) properties of the graph as seen from the vantage point of the vertex being currently occupied. From this perspective, a good fit between the logic and the game also means that the logic should not be excessively expressive: i.e., it should not be possible to express (too many) global properties that have no natural counterpart in the poison game.

Since, by poisoning a vertex in a graph, Poisoner is basically marking that vertex, signalling in this way that it is no longer accessible for Traveller, memory logics [Areces, 2007; Mera, 2009] seem a natural starting point, for they have the ability to store states into a memory. In particular, in this paper we consider three memory logics of decreasing expressive power but increasing fit with the poison game. The first one is the basic memory logic restricted to the initial class of models with an empty memory, which we denote as $\mathcal{ML}_\emptyset$ (see [Areces et al., 2011]). The other two are syntactic fragments of $\mathcal{ML}_\emptyset$, which we respectively denote as $\mathcal{PML}$ and $\mathcal{PSL}$.

We analyse the expressive power of these languages, define notions of bisimulation that are appropriate for $\mathcal{PML}$ and $\mathcal{PSL}$, respectively, and prove that $\mathcal{PSL}$ is strictly less expressive than $\mathcal{PML}$ and $\mathcal{PML}$ is strictly less expressive than $\mathcal{ML}_\emptyset$. We also show that model checking for both $\mathcal{PML}$ and $\mathcal{PSL}$ is PSPACE-complete. In establishing this, we also provide a lower bound on the complexity of the poison game itself: determining the existence of a winning strategy for Traveller is shown to be PSPACE-hard. It is known that the satisfiability problem for $\mathcal{ML}_\emptyset$ is undecidable. Here, we prove that the satisfiability problem for $\mathcal{PML}$ is undecidable, too. We leave it as an open question whether $\mathcal{PSL}$ satisfiability is decidable or not.

The present paper extends and improves on some previous work [Mierzewski and Zaffora Blando, 2016] by the first two authors. The logic $\mathcal{PML}$ was also independently investigated by Grossi and Rey [2019] in the context of abstract argumentation theory (specifically, to study *credulously admissible arguments*[3]).

2 $\mathcal{ML}_\emptyset$

The simplest memory logic extends the basic modal language with two operators: $\textcircled{k}$ and $\textcircled{r}$. A model for this logic is a tuple $\mathcal{M} = (W, R, V, M)$, where (W, R, V) is a standard relational structure, and $M \subseteq W$ is the *memory* of the model. The semantics of the new operators is then given by:

$$\begin{array}{lll} \mathcal{M}, w \models \textcircled{k} & \text{iff } w \in M, & \\ \mathcal{M}, w \models \textcircled{r}\varphi & \text{iff } \mathcal{M}[w], w \models \varphi & \text{where } \mathcal{M}[w] = (W, R, V, M \cup \{w\}). \end{array}$$

The $\textcircled{k}$ operator allows to check whether the current state has been memorised, while the $\textcircled{r}$ operator elicits the memorisation of the current state and the subsequent evaluation of φ there.

To model the poison game, it is reasonable to focus on the class of initial models where $M = \emptyset$, as no vertices are poisoned at the beginning of the game. We will refer to the memory logic restricted to this class of initial models as $\mathcal{ML}_\emptyset$. In this language, we can use formulas of the form $\Diamond\textcircled{r}\varphi$ to model Poisoner's moves and formulas of the form $\Diamond\neg\textcircled{k}$ to model Traveller's moves.

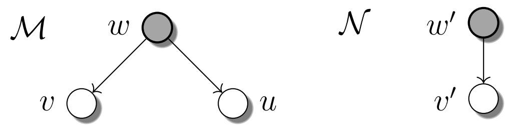
The logic $\mathcal{ML}_\emptyset$ is very expressive. For instance, we can express the property that Traveller can survive at least n rounds of a poison game by means of the following inductive scheme:

$$\rho_1 := \Box\textcircled{r}(\Box\bot \vee \Diamond\neg\textcircled{k})$$

$$\rho_n := \Box\textcircled{r}(\Box\bot \vee \Diamond(\neg\textcircled{k} \wedge \rho_{n-1}))$$

We can also express the property that the point of evaluation has a successor that has itself as its only successor via the formula $\Diamond\textcircled{r}(\Diamond\textcircled{k} \wedge \Box\textcircled{k})$. This implies that $\mathcal{ML}_\emptyset$ does not have the tree model property: the formula $\Diamond\textcircled{r}(\Diamond\textcircled{k} \wedge \Box\textcircled{k})$ cannot be satisfied at the root of a tree.

The property of having n non-poisoned successors, on the other hand, is not expressible within $\mathcal{ML}_\emptyset$. To see why, consider the models $\mathcal{M} = (W, R, V, \emptyset)$ and $\mathcal{N} = (W', R', V', \emptyset)$ below, where $V(q) = V'(q) = \emptyset$ for all proposition letters q.



[3]See, for instance, [Vreeswijk and Prakken, 2000].

Note that it is possible to define a notion of bisimulation that is suitable for $\mathcal{ML}_\emptyset$ (see [Areces et al., 2011, Definition 3.4]). Now, although state w from model $\mathcal{M}$ has two non-poisoned successors, while state w' from model $\mathcal{N}$ only has one, $(\mathcal{M}, w)$ and $(\mathcal{N}, w')$ are $\mathcal{ML}_\emptyset$ bisimilar. The basic intuition behind this observation is that, in order to mark the successors of w and w' via the ⓡ operator, one has to first move there via the standard $\Diamond$ modality. But then, from the perspective of states v, u and v', models $\mathcal{M}$ and $\mathcal{N}$ are completely indistinguishable.

A first worry raised by the use of $\mathcal{ML}_\emptyset$ to model the poison game is that the ⓡ operator does not, by itself, naturally correspond to Poisoner's moves, for it allows one to memorise the current state, while Poisoner must always poison a *successor* of the current state. In other words, $\mathcal{ML}_\emptyset$ seems to be too expressive for the purpose of faithfully modelling the poison game.

In addition, $\mathcal{ML}_\emptyset$ is not computationally well-behaved:

Theorem 2.1 (Areces et al. [2011]). *The satisfiability problem for $\mathcal{ML}_\emptyset$ is undecidable.*

In light of these considerations, a natural question is whether we can find a logic that is closer to the poison game and with a better computational behaviour.

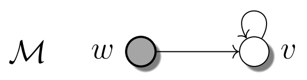
3 $\mathcal{PML}$

As our first attempt, we shall focus on the fragment of $\mathcal{ML}_\emptyset$ that extends the basic modal language with two operators, ⓟ and $\langle \mathtt{p} \rangle$, respectively defined as ⓟ $\leftrightarrow$ ⓚ and $\langle \mathtt{p} \rangle\varphi \leftrightarrow \Diamond$ⓡ$\varphi$. We will refer to this logic as $\mathcal{PML}$.[4]

Expressive power. Since the $\langle \mathtt{p} \rangle$ operator forces the poisoning move to occur one step ahead of the point of evaluation, the following formula is a simple validity of $\mathcal{PML}$:

$$\langle \mathtt{p} \rangle ⓟ \leftrightarrow \Diamond\top$$

It is also worth noting that $\langle \mathtt{p} \rangle$ and $\Diamond$ agree on formulas that do not contain ⓟ: i.e., $\langle \mathtt{p} \rangle\varphi \leftrightarrow \Diamond\varphi$ is valid when φ is a formula not containing ⓟ. This no longer holds in the presence of ⓟ. For instance, consider the formula $\langle \mathtt{p} \rangle$ⓟ $\leftrightarrow \Diamond$ⓟ and the model $\mathcal{M} = (W, R, V, \emptyset)$ below, where $V(q) = \emptyset$ for all proposition letters q.



[4] We use ⓟ instead of ⓚ simply because the former is more suggestive of the *poison* game. Also note that $\mathcal{PML}$ models are of course exactly the same as $\mathcal{ML}_\emptyset$ models, except that we will write P instead of M to denote the set of poisoned states. $\mathcal{PML}$ stands for *poison memory logic.*

Here, we have that $\mathcal{M}, w \models \langle \mathtt{p} \rangle ⓟ$, but $\mathcal{M}, w \not\models \Diamond ⓟ$.

As in the case of $\mathcal{ML}_\emptyset$, the following inductive scheme allows to express the property that Traveller can survive at least n rounds of a poison game:

$$\rho_1 := [\mathtt{p}](\Box\bot \vee \Diamond\neg ⓟ)$$

$$\rho_n := [\mathtt{p}](\Box\bot \vee \Diamond(\neg ⓟ \wedge \rho_{n-1}))$$

Note that the $\mathcal{ML}_\emptyset$ formula $\Diamond ⓡ (\Diamond ⓚ \wedge \Box ⓚ)$ from the previous section—which, as we saw, forces non-tree models in $\mathcal{ML}_\emptyset$—falls within $\mathcal{PML}$ (we can write it as $\langle \mathtt{p} \rangle (\Diamond ⓟ \wedge \Box ⓟ)$). Hence, $\mathcal{PML}$ lacks the tree model property, too.

It is possible to define a notion of bisimulation that fits $\mathcal{PML}$ by adding the following clauses to the standard ones for the basic modal language. Let Z below denote a bisimulation between models $\mathcal{M} = (W, R, V, P)$ and $\mathcal{N} = (W', R', V', P')$:

Non-empty: there are $w \in W$ and $w' \in W'$ with $(P, w)Z(P', w')$;

Agree: if $(S, u)Z(S', u')$, then

(1) $\mathcal{M}, u \models q$ if and only if $\mathcal{N}, u' \models q$ for any proposition letter q, and
(2) $u \in S$ if and only if $u' \in S'$;

Zig$_{\langle \mathtt{p} \rangle}$: if $(S, u)Z(S', u')$ and there exists $v \in W$ with $(u, v) \in R$, then there exists $v' \in W'$ with $(u', v') \in R'$ and $(S \cup \{v\}, v)Z(S' \cup \{v'\}, v')$;

Zag$_{\langle \mathtt{p} \rangle}$: if $(S, u)Z(S', u')$ and there exists $v' \in W'$ with $(u', v') \in R'$, then there exists $v \in W$ with $(u, v) \in R$ and $(S \cup \{v\}, v)Z(S' \cup \{v'\}, v')$.

In the **Agree**, **Zig$_{\langle \mathtt{p} \rangle}$** and **Zag$_{\langle \mathtt{p} \rangle}$** clauses above, we use S and S' instead of P and P' because the $\langle \mathtt{p} \rangle$ modality allows to add new states to the initial memories of the models $\mathcal{M}$ and $\mathcal{N}$.

This notion of $\mathcal{PML}$ bisimulation is correct, in that it implies $\mathcal{PML}$ equivalence:

Proposition 3.1. *Let $\mathcal{M} = (W, R, V, P)$ and $\mathcal{N} = (W', R', V', P')$ be two $\mathcal{PML}$ models, $w \in W$ and $w' \in W'$. If Z is a bisimulation linking (P, w) and (P', w'), then, for any $\varphi \in \mathcal{PML}$,*

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{N}, w' \models \varphi.$$

Proof. The proof is by induction on φ and the only non-standard cases are the ones involving ⓟ and $\langle \mathtt{p} \rangle$. First of all, we have that

$$\begin{aligned} \mathcal{M}, w \models ⓟ &\text{ iff } w \in P && \text{by the semantics of ⓟ,} \\ &\text{ iff } w' \in P' && \text{by the } \textbf{Agree} \text{ condition,} \\ &\text{ iff } \mathcal{N}, w' \models ⓟ && \text{by the semantics of ⓟ.} \end{aligned}$$

Now, suppose $\mathcal{M}, w \models \langle \mathtt{p} \rangle \varphi$. Then, there is $v \in W$ with $(w, v) \in R$ and $\mathcal{M}[v], v \models \varphi$. Since $(P, w)Z(P', w')$ and there is $v \in W$ with $(w, v) \in R$, the $\mathbf{Zig}_{\langle \mathtt{p} \rangle}$ condition gives us that there exists $v' \in W'$ with $(w', v') \in R'$ and $(P \cup \{v\}, v)Z(P' \cup \{v'\}, v')$. Now, since $(P \cup \{v\}, v)Z(P' \cup \{v'\}, v')$ and $\mathcal{M}[v], v \models \varphi$, the induction hypothesis implies that $\mathcal{N}[v'], v' \models \varphi$. Hence, $\mathcal{N}, w' \models \langle \mathtt{p} \rangle \varphi$. The other direction is analogous, except that it relies on the $\mathbf{Zag}_{\langle \mathtt{p} \rangle}$ condition. $\square$

Now, how does $\mathcal{PML}$ compare with $\mathcal{ML}_\emptyset$ in terms of expressive power? We will show that $\mathcal{PML}$ is strictly less expressive than $\mathcal{ML}_\emptyset$.

Definition 3.2. *Let $\mathcal{L}$ and $\mathcal{L}'$ be two logics. We say that $\mathcal{L}'$ is* at least as expressive as *$\mathcal{L}$ (in symbols, $\mathcal{L} \leq \mathcal{L}'$) if there is a translation $\mathsf{T} : \mathcal{L} \to \mathcal{L}'$ such that, for every model $\mathcal{M}$, every w in $\mathcal{M}$ and every $\varphi \in \mathcal{L}$,*

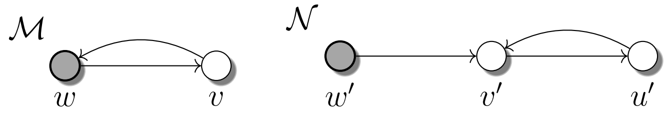
$$\mathcal{M}, w \models_{\mathcal{L}} \varphi \textit{ iff } \mathcal{M}, w \models_{\mathcal{L}'} \mathsf{T}(\varphi),$$

where $\mathcal{M}$ is seen as an $\mathcal{L}$ model on the left-hand side and as an $\mathcal{L}'$ model on the right-hand side, and, in each case, we use the appropriate semantic relation ($\models_{\mathcal{L}}$ and $\models_{\mathcal{L}'}$, respectively). Logic $\mathcal{L}'$ is strictly more expressive *than logic $\mathcal{L}$ (in symbols, $\mathcal{L} < \mathcal{L}'$) if $\mathcal{L} \leq \mathcal{L}'$ but $\mathcal{L}' \not\leq \mathcal{L}$.*

We will appeal to the notion of $\mathcal{PML}$ bisimulation defined above to show that $\mathcal{PML}$ is indeed strictly less expressive than $\mathcal{ML}_\emptyset$:

Proposition 3.3. *$\mathcal{PML} < \mathcal{ML}_\emptyset$.*

Proof. Since $\mathcal{PML}$ is a syntactic fragment of $\mathcal{ML}_\emptyset$, we trivially have that $\mathcal{PML} \leq \mathcal{ML}_\emptyset$. To show that $\mathcal{ML}_\emptyset \not\leq \mathcal{PML}$, consider the models $\mathcal{M} = (W, R, V, \emptyset)$ and $\mathcal{N} = (W', R', V', \emptyset)$ below, where $V(q) = V(q') = \emptyset$ for all proposition letters q.



We have that $(\mathcal{M}, w)$ and $(\mathcal{N}, w')$ are $\mathcal{PML}$ bisimilar. However, $\mathcal{M}, w \models_{\mathcal{ML}_\emptyset} ⓡ\Diamond\Diamond ⓚ$, while $\mathcal{N}, w' \not\models_{\mathcal{ML}_\emptyset} ⓡ\Diamond\Diamond ⓚ$. $\square$

Since it is a fragment of $\mathcal{ML}_\emptyset$, the logic $\mathcal{PML}$ is evidently a fragment of first-order logic [Areces, 2007; Mera, 2009]. A direct translation into first-order logic is given below.

Proposition 3.4. *There is an effective meaning-preserving translation from $\mathcal{PML}$ into first-order logic.*

Proof. We define a translation ST_y^X from $\mathcal{PML}$ formulas to first-order formulas, where y is a variable and X a finite set of variables:

$$\begin{aligned}
\mathsf{ST}_y^X(p) &= Py \\
\mathsf{ST}_y^X(\neg\varphi) &= \neg\mathsf{ST}_y^X(\varphi) \\
\mathsf{ST}_y^X(\varphi \vee \psi) &= \mathsf{ST}_y^X(\varphi) \vee \mathsf{ST}_y^X(\psi) \\
\mathsf{ST}_y^X(\Diamond\varphi) &= \exists z\Big(Ryz \wedge \mathsf{ST}_z^X(\varphi)\Big) \\
\mathsf{ST}_y^X(\langle \mathsf{p}\rangle\varphi) &= \exists z\Big(Ryz \wedge \mathsf{ST}_z^{X\cup\{z\}}(\varphi)\Big) \\
\mathsf{ST}_y^X(\textcircled{\mathsf{p}}) &= \bigvee_{x\in X} y = x
\end{aligned}$$

Given a pointed model $(\mathcal{M}, w)$ and a finite set $D = \{d_1, ..., d_n\}$ of states in $\mathcal{M}$, it is easy to prove by induction on $\mathcal{PML}$ formulas φ and sets of variables $X = \{x_1, ..., x_n\}$ that

$$(\mathcal{M}, w) \models \varphi \text{ iff } \mathcal{M} \models \mathsf{ST}_y^X(\varphi)[y := w, x_1 := d_1, ..., x_n := d_n].$$

(Note that the set X in the above translation is used to keep track of the states that have already been poisoned.) □

Model checking. We now show that model checking for $\mathcal{PML}$ is PSPACE-complete. That it is at most PSPACE is established by the first-order translation given in Proposition 3.4, which results in a polynomial increase in length and can be done in polynomial time (alternatively, recall that $\mathcal{PML}$ is a fragment of the basic memory logic $\mathcal{ML}_\emptyset$). For the lower bound, we provide a polynomial-time reduction from the true quantified Boolean formula (QBF) problem. Given a QBF φ, we build a graph in which Traveller has a winning strategy in the poison game if and only if φ is true, and for which the existence of a winning strategy is expressible by a $\mathcal{PML}$ formula (where the size of the graph and of the formula increase only linearly in the size of φ). This also gives a reduction of QBF to the problem of determining the existence of a winning strategy in the poison game itself, and it thus provides a lower complexity bound for the game. This method of reduction by games for modal logics originates in [Löding and Rohde, 2003a] and [Rohde, 2005], where it was used in the study of the sabotage game: the method is worth noting, as it can yield simple and useful model checking results in dynamic modal logics more generally (see [van Benthem et al., 2019] for another application).

Theorem 3.5. *Model checking for $\mathcal{PML}$ is* PSPACE*-complete.*

Proof. We reduce QBF to model checking for $\mathcal{PML}$. Recall that a *fully quantified*

Boolean formula (QBF) is one of the form

$$Q_1 x_1 \dots Q_n x_n \bigwedge_{1 \leq i \leq m} C_i,$$

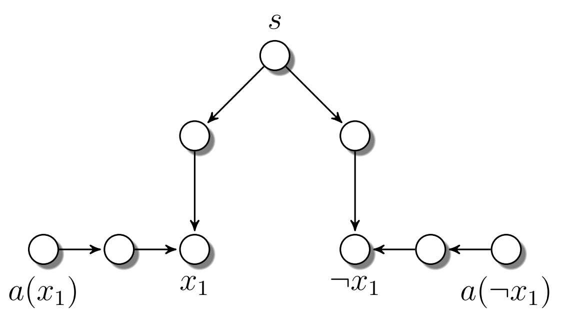
where each $Q_j \in \{\exists, \forall\}$ and each C_i is a disjunction of literals $\pm x_j$ $(j \leq n)$. Without loss of generality, we can assume that $Q_1 = \exists$. The QBF problem consists in determining whether the formula is true when the quantifiers range over truth-value assignments to the variables x_j. For each given QBF φ, we construct a pointed model $(\mathcal{M}_\varphi, s)$ and a formula Γ_φ such that φ is true if and only if $\mathcal{M}_\varphi, s \models \Gamma_\varphi$. The model $\mathcal{M}_\varphi$ is constructed from basic modules as depicted in Figure 1. We begin with the *initial module* (Figure 1(a)), which contains the evaluation point s and corresponds to the initial quantifier $\exists x_1$. We then concatenate subsequent modules—each either a $\forall x_j$*-module* (Figure 1(c)) or a $\exists x_j$*-module* (Figure 1(b))—for consecutive variables x_j, in the order corresponding to the order of quantifiers Q_j in φ.

Concatenating modules here simply amounts to identifying the last row of vertices in each module (with labels $a(x_j), x_j, \neg x_j \dots$) with the first row of vertices of the next module: that is, for the top nodes of the j-th module we take the end nodes of the $(j-1)$-th module. Once all n quantifier modules have been added in this fashion, we append the *final verification module* (Figure 1(d)) in the same way. Each clause C_i in φ is associated with exactly one node c_i in the verification module—its *clause vertex*. Each clause vertex c_i has an outgoing edge to all and only the vertices $a(\ell)$, where $\ell = \pm x_j$ is a literal that makes C_i true.

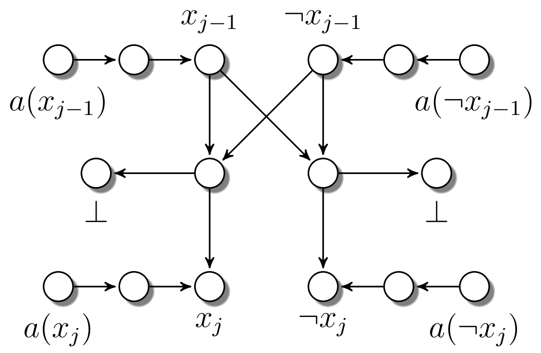
Consider now the poison game played on this graph. Traveller begins the game at s and makes the first move. By choosing to go left or right, she forces Poisoner to poison exactly one of the vertices labelled x_1 or $\neg x_1$. The same holds at each $\exists x_j$-module: Traveller has the power to determine which of x_j or $\neg x_j$ will be poisoned. Similarly, at $\forall x_j$-modules, Traveller makes the only available first move, after which it is Poisoner who chooses which one between x_j and $\neg x_j$ to poison (note that Poisoner never selects the endpoints, marked by $\perp$, since endpoints in the graph are winning positions for Traveller).

Observation. Traveller *has a winning strategy for the poison game on* $(\mathcal{M}_\varphi, s)$ *if and only if the initial* QBF *formula* φ *is true.*

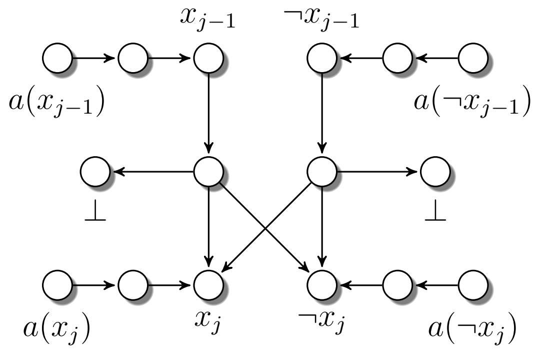
In this setting, Traveller winning the poison game is equivalent to the $\exists$-player winning the *formula game* on QBF formulas (see [Sipser, 2012]). A useful heuristic for thinking about the game is as follows. Each passage through the graph all the way until the final verification module corresponds to selecting a valuation: for each j, exactly one of the vertices marked with x_j or $\neg x_j$ is poisoned. The non-poisoned vertices correspond to the selected valuation. Traveller aims at a final valuation that



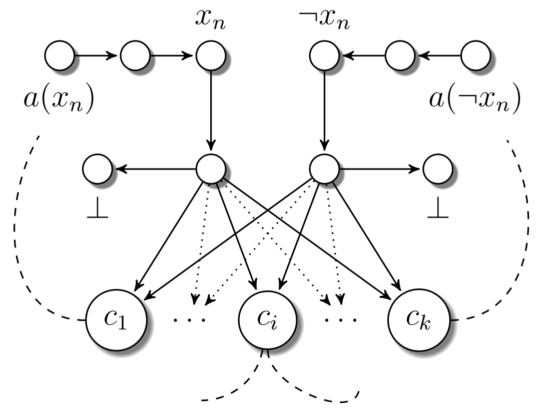
(a) Initial module.



(b) $\exists x_j$-module.



(c) $\forall x_j$-module.



(d) Final verification module.

Figure 1: The four basic modules. In (b), (c) and (d), the top nodes labelled by x_{j-1} and $\neg x_{j-1}$ (x_n and $\neg x_n$ in (d)) are the end nodes of the previous module. In (d), each clause vertex c_i has an outgoing edge to all and only the vertices $a(\ell)$, where ℓ is a literal $\pm x_i$ that makes clause C_i true.

makes the QBF formula true, while Poisoner tries to make the formula false. If Traveller wants $\pm x_j$ to be true at a $\exists x_j$-module, she forces Poisoner to pass through $\mp x_j$, so that the vertex $\pm x_j$ is spared. Similarly, if Poisoner wishes to make $\pm x_j$ false at a $\forall x_j$-module, she makes sure to poison it. At the verification module, Poisoner selects a clause node c_i: at this point, either there is some vertex $a(\pm x_j)$ accessible from c_i for some non-poisoned $\pm x_j$ that makes C_i true, or $\pm x_j$ is poisoned for every accessible vertex $a(\pm x_j)$. In the former case, Traveller can move from c_i to such a vertex $a(\pm x_j)$, where $\pm x_j$ is not poisoned. After any three more moves in the game, Traveller then finds herself, at the beginning of her turn, at a point that sees an endpoint (marked by $\perp$ on the graph). She travels to the endpoint and wins the game. If, on the other hand, $\pm x_j$ is poisoned for every $\pm x_j$ that makes C_i true, Traveller is compelled to move to some $a(\pm x_j)$ where the vertex $\pm x_j$ is poisoned. At her next turn, Traveller loses the game, since she is forced to move to a poisoned vertex.

Lastly, we make sure that the existence of a winning strategy is expressible by a $\mathcal{PML}$ formula. Recall the formula ρ_k expressing survival for at least k rounds, given by the scheme

$$\rho_1 := [\mathtt{p}](\Box\bot \vee \Diamond\neg\textcircled{p})$$

$$\rho_k := [\mathtt{p}](\Box\bot \vee \Diamond(\neg\textcircled{p} \wedge \rho_{k-1}))$$

Given a QBF φ with n variables, take the $\mathcal{PML}$ formula

$$\Gamma_\varphi = \Diamond(\neg\textcircled{p} \wedge \rho_{n+3})$$

Is is easy to see that Traveller has a winning strategy if and only if $\mathcal{M}_\varphi, s \models \Gamma_\varphi$. The formula states that Traveller can make a first move to a non-poisoned point, after which she can survive for $n+3$ rounds in the poison game. This is equivalent to Traveller winning the poison game. Note that it takes exactly n rounds of the game to reach the last row of vertices from which the clause nodes are accessible. The $(n+1)$-st round starts with Poisoner selecting a clause vertex in the final verification module, and Traveller responding by selecting some accessible vertex $a(\ell)$. If all accessible points $a(\ell)$ have the vertex ℓ poisoned, then Traveller loses in the next, $(n+2)$-nd round, and Γ_φ fails. Otherwise, Traveller survives for one more round (and so Γ_φ holds), and now she has a guaranteed victory by moving to an endpoint in the $(n+3)$-rd round. The reduction is polynomial in the size of φ. Note that the size of the model $\mathcal{M}_\varphi$ grows linearly in the number of variables, and so does the size of Γ_φ: when φ has n variables and m clauses, we have that $|\mathcal{M}_\varphi| \leq \alpha(n+1)+m$, where α is the maximum size of a quantifier module, and the number of edges is bounded above by $\beta(n+1)+m(n+2)$, where β is the maximum number of edges is a non-final module. $\Box$

The construction just given reduces the truth of a QBF not only to the truth of a $\mathcal{PML}$ formula in a model, but also to the existence of a winning strategy in a poison game. Thus, the argument above gives us an immediate corollary:

Proposition 3.6. *The existence of a winning strategy for* Traveller *in the poison game is* PSPACE*-hard.*

Thus, what falls out of this simple analysis of model checking is a lower bound on the complexity of testing for the existence of a winning strategy in the poison game. In fact, Zhang [2019] has independently shown that this problem is PSPACE-complete.

$$\varphi = \exists x_1 \forall x_2 \exists x_3 (C_1 \wedge C_2 \wedge C_3)$$
where $C_1 = x_1 \vee x_2 \vee x_3$, $C_2 = \neg x_1 \vee \neg x_3$, and $C_3 = \neg x_2 \vee \neg x_3$

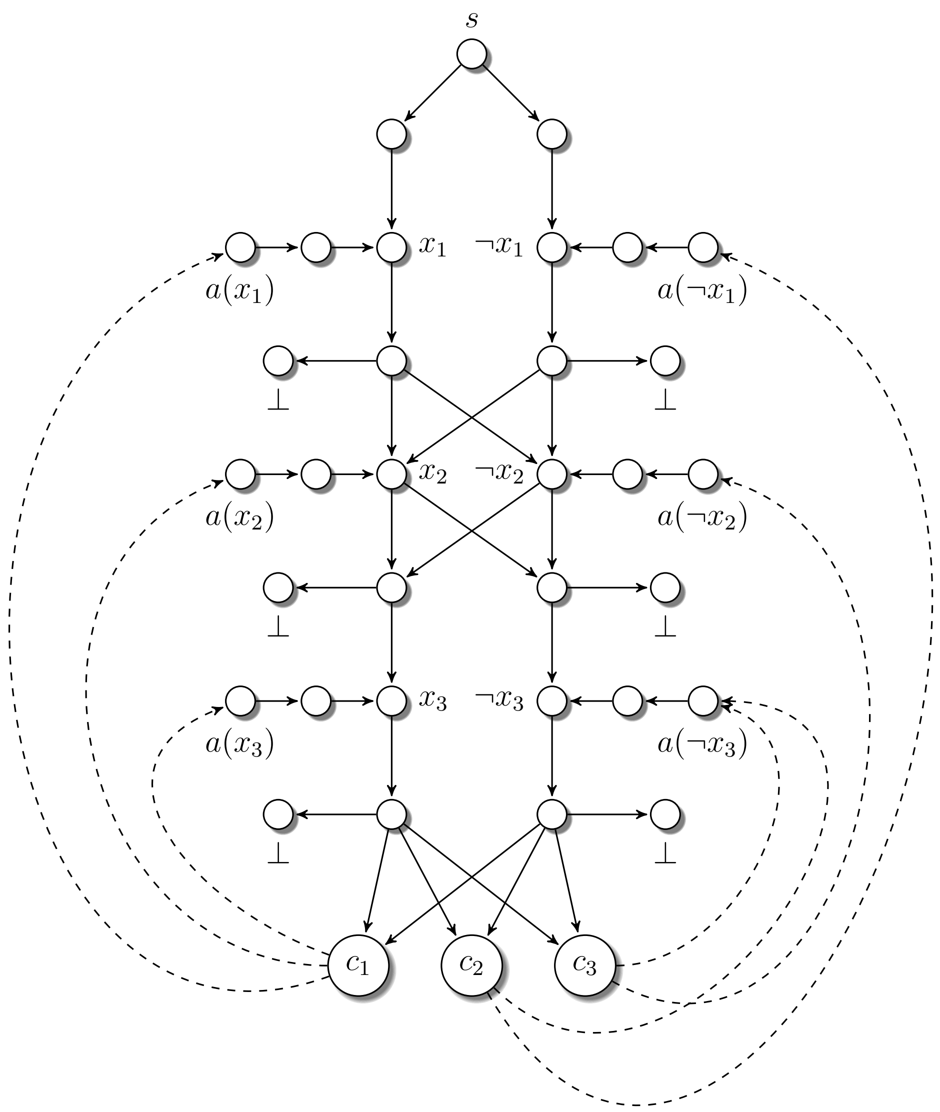


Figure 2: An example. At each $\exists x_j$-module, Traveller forces Poisoner to poison a node labelled by the literal $\pm x_j$ of Traveller's choice. Each $\forall x_j$-module, on the other hand, leaves the choice to Poisoner. In the final verification module, Poisoner forces Traveller into some clause node c_i. For each literal $\pm x_j$ that makes the corresponding clause C_i true, Traveller can move to the node $a(\pm x_j)$ above. In this example, Traveller has a winning strategy which ensures that she can reach an endpoint $\perp$, where she wins.

Satisfiability. We conclude this section by considering the satisfiability problem for $\mathcal{PML}$. In spite of being strictly less expressive than $\mathcal{ML}_\emptyset$ (Proposition 3.3), the satisfiability problem for $\mathcal{PML}$ is undecidable, just like the one for $\mathcal{ML}_\emptyset$.

We begin by showing that $\mathcal{PML}$ does not have the finite model property. To do so, we make use of the 'spy-point technique', first introduced by Blackburn and Seligman [1995] in the context of *hybrid logics*.

Theorem 3.7. *$\mathcal{PML}$ lacks the finite model property.*

Proof. Consider the following formulas:

$$
\begin{array}{rl}
(\mathit{Back}) & q \wedge \neg s \wedge \Diamond\top \wedge \Box(\neg q \wedge s \wedge \Diamond\top \wedge \Box(\neg q \wedge \neg s \wedge \Box\neg q)) \wedge [\mathrm{p}]\Box\Diamond\textcircled{p} \wedge [\mathrm{p}]\Box\Box(s \to \textcircled{p}) \\
(\mathit{Spy}) & [\mathrm{p}][\mathrm{p}][\mathrm{p}]\big(\neg s \to \Diamond(s \wedge \textcircled{p} \wedge \Diamond(\neg s \wedge \textcircled{p} \wedge \Box(\neg s \to \neg\textcircled{p})))\big) \\
(\mathit{Irr}) & \Box[\mathrm{p}]\Box\neg\textcircled{p} \\
(\mathit{Succ}) & \Box\Box\Diamond\neg s \\
(\mathit{No\text{-}3cyc}) & [\mathrm{p}][\mathrm{p}][\mathrm{p}]\big(\neg s \to [\mathrm{p}](\neg s \to \Box(\neg s \to \neg\textcircled{p}))\big) \\
(\mathit{Trans}) & [\mathrm{p}][\mathrm{p}]\Box\big(\neg s \to [\mathrm{p}]\big(\neg s \to \Diamond(s \wedge \textcircled{p} \wedge \Diamond(\neg s \wedge \textcircled{p} \wedge \Diamond(\neg s \wedge \textcircled{p})))\big)\big)
\end{array}
$$

Now, let *Inf* be the formula (*Back* $\wedge$ *Spy* $\wedge$ *Irr* $\wedge$ *Succ* $\wedge$ *No-3cyc* $\wedge$ *Trans*). We are going to show that, for any $\mathcal{PML}$ model $\mathcal{M} = (W, R, V, \emptyset)$ and any $w \in W$, if $\mathcal{M}, w \models \mathit{Inf}$, then W must be infinite. So, suppose that $\mathcal{M}, w \models \mathit{Inf}$. Then, by (*Back*), we know that w has a successor, call it v, that satisfies s. Any such successor of w will behave as a spy point. Now, (*Back*) ensures that w does not see itself, and that no spy point can access w. Moreover, (*Back*) guarantees that (i) spy points have at least one successor and that no spy point can see itself, (ii) all successors of a spy point can see this spy point back, and (iii) all successors of a spy point see exactly one spy point. (*Spy*) then ensures that all successors of a successor of a spy point see this spy point back and are directly seen by it. By (*Irr*), all successors of a spy point are irreflexive and, by (*Succ*), all successors of a spy point see a non-spy point (which cannot be w). (*Back*), (*Spy*), (*Irr*) and (*No-3cyc*) together ensure that there are no 2-cycles and no 3-cycles among the successors of a spy point. Finally, (*Trans*)—together with (*Back*), (*Spy*), (*Irr*), (*Succ*) and (*No-3cyc*)—guarantees that the relation R is transitive on the set of successors of a spy point. Now, consider the spy point v: it follows from the above reasoning that its set of successors is an unbounded strict partial order, which entails that W is infinite. Lastly, the model depicted in Figure 3 establishes that (*Inf*) is indeed satisfiable. $\Box$

To prove undecidability, we will encode the $\omega \times \omega$ tiling problem within our language. In doing so, we shall once again employ a spy-point argument. To streamline

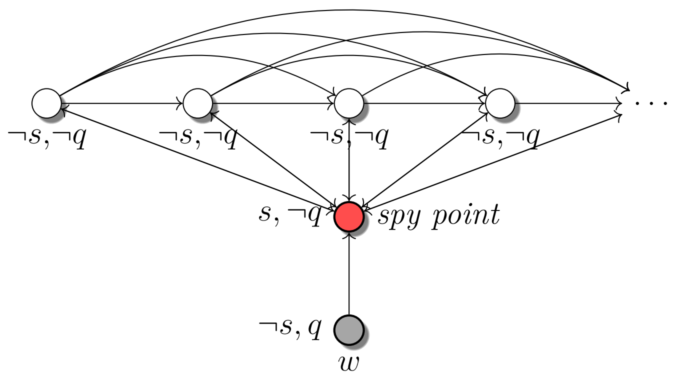


Figure 3: A model for *Inf*.

the presentation of the proof, we will at first help ourselves to three accessibility relations: R_s, R_u and R_r—where R_s will be used to model transitions from/to the spy point to/from the grid points, R_u for upward transitions along the grid, and R_r for rightward transitions along the grid. We will thus show that the satisfiability problem for $\mathcal{PML}(\Diamond_s, \Diamond_u, \Diamond_r)$ is undecidable. After presenting this argument, we will then explain how the proof of Theorem 3.8 below can be turned into a proof of the fact that the satisfiability problem for $\mathcal{PML}$, where only one accessibility relation is available, is undecidable, too.

Theorem 3.8. *The satisfiability problem for $\mathcal{PML}(\Diamond_s, \Diamond_u, \Diamond_r)$ is undecidable.*

Proof. Let $\mathcal{T} = \{T_1, ..., T_n\}$ be a finite set of tile types. Given a tile type T_i, $u(T_i), r(T_i)$, $d(T_i)$ and $l(T_i)$ will represent the colours of the upper, right, lower and left edges of T_i, respectively. For each tile type T_i, we fix a proposition letter t_i that is going to encode T_i. We now define a formula $\varphi_\mathcal{T}$ such that $\varphi_\mathcal{T}$ is satisfiable if and only if $\mathcal{T}$ tiles $\omega \times \omega$. Consider the following formulas:

(*Back*) $q \wedge \neg s \wedge \Diamond_s \top \wedge \Box_s(\neg q \wedge s \wedge \Diamond_s \top \wedge \Box_s(\neg q \wedge \neg s)) \wedge [\mathtt{p}]_s \Box_s \Diamond_s (s \wedge \textcircled{\texttt{p}}) \wedge$
$[\mathtt{p}]_s \Box_s \Box_s (s \wedge \textcircled{\texttt{p}})$

(*Spy*) $[\mathtt{p}]_s[\mathtt{p}]_s[\mathtt{p}]_u\Big(\neg q \wedge \neg s \wedge \Diamond_s(s \wedge \textcircled{\texttt{p}} \wedge \Diamond_s(\textcircled{\texttt{p}} \wedge \Box_u \neg \textcircled{\texttt{p}} \wedge \Box_r \neg \textcircled{\texttt{p}}))\Big)$
$[\mathtt{p}]_s[\mathtt{p}]_s[\mathtt{p}]_r(\neg q \wedge \neg s \wedge \Diamond_s(s \wedge \textcircled{\texttt{p}} \wedge \Diamond_s(\textcircled{\texttt{p}} \wedge \Box_r \neg \textcircled{\texttt{p}} \wedge \Box_u \neg \textcircled{\texttt{p}}))\Big)$

(*Grid*) $\Box_s \Box_s (\Diamond_u \top \wedge \Diamond_r \top)$

(*Func*) $\Box_s [\mathtt{p}]_s [\mathtt{p}]_u \Diamond_s \Big(s \wedge \Diamond_s(\textcircled{\texttt{p}} \wedge \Diamond_u \textcircled{\texttt{p}} \wedge \Box_u \textcircled{\texttt{p}})\Big)$
$\Box_s [\mathtt{p}]_s [\mathtt{p}]_r \Diamond_s \Big(s \wedge \Diamond_s(\textcircled{\texttt{p}} \wedge \Diamond_r \textcircled{\texttt{p}} \wedge \Box_r \textcircled{\texttt{p}})\Big)$

$$\begin{array}{rl}
(\textit{UR-no-Cycle}) & \Box_s[\mathtt{p}]_s[\mathtt{p}]_u\Box_r\neg\textcircled{\mathtt{p}} \wedge \Box_s[\mathtt{p}]_s[\mathtt{p}]_r\Box_u\neg\textcircled{\mathtt{p}} \\
(\textit{URU-no-Cycle}) & \Box_s[\mathtt{p}]_s[\mathtt{p}]_u[\mathtt{p}]_r\Box_u\neg\textcircled{\mathtt{p}} \\
(\textit{Confl}) & \Box_s[\mathtt{p}]_s\langle\mathtt{p}\rangle_u\langle\mathtt{p}\rangle_r\Diamond_s\Diamond_s\Big(\textcircled{\mathtt{p}} \wedge \Box_r\neg\textcircled{\mathtt{p}} \wedge \Diamond_u\textcircled{\mathtt{p}} \wedge \Diamond_r\Diamond_u(\textcircled{\mathtt{p}} \wedge \Box_r\neg\textcircled{\mathtt{p}})\Big) \\
(\textit{Unique}) & \Box_s\Box_s\left(\bigvee_{1\leq i\leq n} t_i \wedge \bigwedge_{1\leq i<j\leq n}(t_i \to \neg t_j)\right) \\
(\textit{Vert}) & \Box_s\Box_s\bigwedge_{1\leq i\leq n}\left(t_i \to \Diamond_u \bigvee_{1\leq j\leq n, u(T_i)=d(T_j)} t_j\right) \\
(\textit{Horiz}) & \Box_s\Box_s\bigwedge_{1\leq i\leq n}\left(t_i \to \Diamond_r \bigvee_{1\leq j\leq n, r(T_i)=l(T_j)} t_j\right)
\end{array}$$

Now, let $\varphi_{\mathcal{T}}$ be the conjunction of all of the formulas above.

($\Rightarrow$) Suppose that $\mathcal{M}, w \models \varphi_{\mathcal{T}}$, for some $\mathcal{PML}(\Diamond_s, \Diamond_u, \Diamond_r)$ model $\mathcal{M} = (W, R_s, R_u, R_r, V, \emptyset)$ and $w \in W$. The formula (*Back*) and the two (*Spy*) formulas ensure that w has a successor that is a spy point via the relation R_s. They also guarantee that (1) w is not a successor of the spy point, (2) w is not a successor of any successor of the spy point, (3) R_s, R_u and R_r are irreflexive, and (4) R_u and R_r are asymmetric. The points that are R_s-accessible from the spy point represent the tiles. The formula (*Grid*)—together with (*Back*) and (*Spy*)—ensures that every tile (i.e., every point accessible from the spy point) has a tile above it, via the R_u relation, and a tile to its right, via the R_r relation. The two (*Func*) formulas—together with (*Back*) and (*Spy*)—on the other hand, guarantee that R_u and R_r are functional: namely, that every tile has at most one tile above it and at most one tile to its right. So, (*Grid*) and (*Func*) together ensure that every tile has exactly one tile above it and exactly one tile to its right. Now, (*UR-no-Cycle*) ensures that no tile can be both above/below and to the left/right of another tile, while (*URU-no-Cycle*) forbids cycles following successive steps of the R_u, R_r and R_u relations, in this order. Together with (*UR-no-Cycle*) and (*URU-no-Cycle*), (*Conf*) then ensures that the tiles are arranged in a grid. Finally, (*Unique*) guarantees that every tile has a unique type, while (*Vert*) and (*Horiz*) ensure that the colours of the tiles match appropriately. It then follows that $\mathcal{M}$ yields a tiling of $\omega \times \omega$.

($\Leftarrow$) For the other direction, suppose that $f : \omega \times \omega \to \mathcal{T}$ is a tiling of $\omega \times \omega$. Let $\mathcal{M}$ be the $\mathcal{PML}(\Diamond_s, \Diamond_u, \Diamond_r)$ model $(\omega \times \omega \cup \{w, v\}, R_s, R_u, R_r, V, \emptyset)$, where the accessibility relations R_s, R_u and R_r are given by

$$\begin{aligned}
R_s &= \{(w, v)\} \cup \{(v, x), (x, v) : x \in \omega \times \omega\} \\
R_u &= \{((n, m), (n, m+1)) : n, m \in \omega\} \\
R_r &= \{((n, m), (n+1, m)) : n, m \in \omega\}
\end{aligned}$$

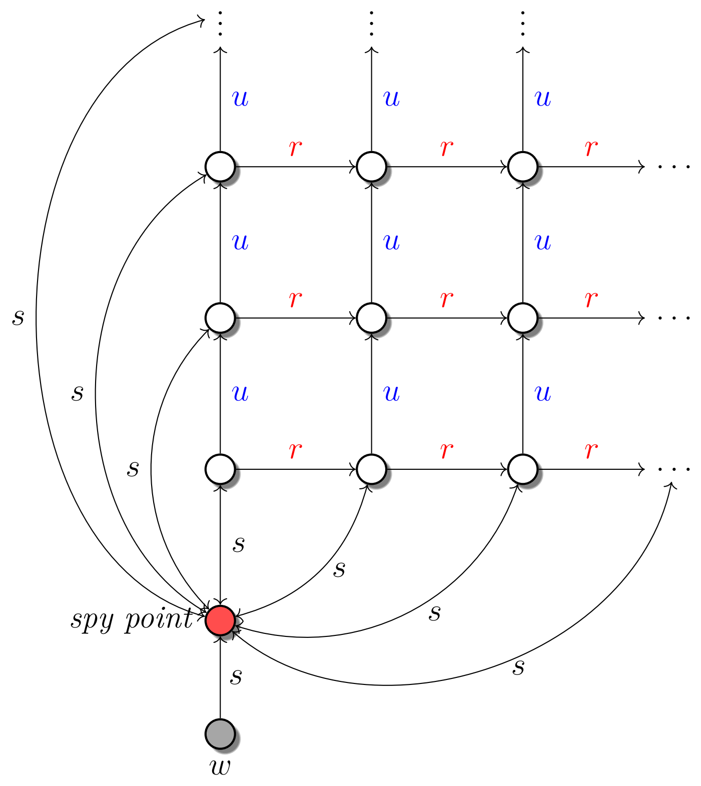


Figure 4: A model for $\varphi_{\mathcal{T}}$: a $\omega \times \omega$ grid with a spy point.

and the valuation V is defined as

$$\begin{aligned}
V(q) &= \{w\} \\
V(s) &= \{v\} \\
V(t_i) &= \{x : x \in \omega \times \omega \text{ and } f(x) = T_i\} \text{ for all } 1 \leq i \leq n \\
V(p) &= \emptyset \text{ for all other proposition letters } p
\end{aligned}$$

The above specification of $\mathcal{M}$ ensures that v is a spy point. By construction, we then have that $\mathcal{M}, w \models \varphi_{\mathcal{T}}$. □

So, $\mathcal{PML}(\Diamond_s, \Diamond_u, \Diamond_r)$ is undecidable. The additional modalities render the encoding less cumbersome, but, as a matter of fact, they are not required for undecidability: an argument analogous to the one given by Hoffmann [2015] can be employed to adapt our proof of Theorem 3.8 to the original $\mathcal{PML}$ language with only one accessibility relation R. The basic idea for such a modification consists in using proposition letters u and r to appropriately encode the relations R_u and R_r.

Theorem 3.9. *The satisfiability problem for $\mathcal{PML}$ is undecidable.*

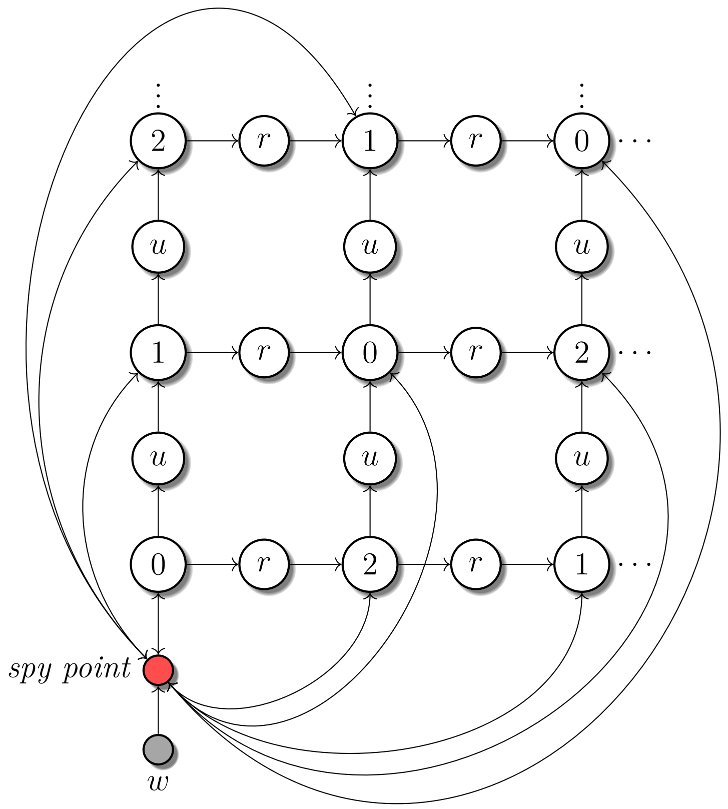


Figure 5: Encoding the grid with a single accessibility relation.

Proof. Consider the set of proposition letters $V := \{0, 1, 2, q, s, r, u\}$. For any $\ell \in V$, let $\mathsf{X}_\ell := \ell \wedge \bigwedge_{v \in V \setminus \{\ell\}} \neg v$ be the formula stating that ℓ holds and all other atomic propositions in V are false. For $i \in \{0, 1, 2\}$, let $\mathsf{n}(i) = i + 1 \pmod 3$ and $\mathsf{e}(i) = i + 2 \pmod 3$. Now, consider the following formulas, where $i \in \{0, 1, 2\}$ and $a \in \{r, u\}$:

$$(\mathit{Grid}_1) \qquad \mathsf{X}_q \wedge \Diamond\top \wedge \Box\Big(\mathsf{X}_s \wedge \Diamond\top \wedge \Box(\mathsf{X}_0 \vee \mathsf{X}_1 \vee \mathsf{X}_2)\Big)$$

$$(\mathit{Grid}_2) \qquad \Box\Box\Big(\Diamond\mathsf{X}_r \wedge \Diamond\mathsf{X}_u \wedge \Box(\mathsf{X}_r \vee \mathsf{X}_u \vee \mathsf{X}_s)\Big)$$

$$(\beta_i) \qquad \Box\Box\bigg(i \rightarrow \Box\Big((r \rightarrow \Diamond\mathsf{e}(i) \wedge \Box\,\mathsf{X}_{\mathsf{e}(i)}) \wedge (u \rightarrow \Diamond\mathsf{n}(i) \wedge \Box\,\mathsf{X}_{\mathsf{n}(i)})\Big)\bigg)$$

$$(\mathit{Back}) \qquad [\mathtt{p}]\Box\Big(\Diamond(s \wedge \textcircled{p}) \wedge \Box(s \rightarrow \textcircled{p})\Big)$$

$$(\mathit{Spy}) \qquad [\mathtt{p}][\mathtt{p}][\mathtt{p}]\bigg(a \rightarrow [\mathtt{p}]\Diamond\Big(s \wedge \textcircled{p} \wedge \Diamond(\textcircled{p} \wedge \Box(a \rightarrow \neg\textcircled{p}))\Big)\bigg)$$

$$(\mathit{Func}) \qquad \Box[\mathtt{p}][\mathtt{p}]\bigg(a \rightarrow [\mathtt{p}]\Diamond\Big(s \wedge \Diamond(\textcircled{p} \wedge \Diamond(\textcircled{p} \wedge a \wedge \Diamond\textcircled{p}) \wedge \Box(a \rightarrow \textcircled{p} \wedge \Box\textcircled{p})\Big)\bigg)$$

We then also add the formulas ($\mathit{UR\text{-}no\text{-}Cycle}^*$), ($\mathit{Conf}^*$), ($\mathit{Unique}^*$), ($\mathit{Vert}^*$) and ($\mathit{Horiz}^*$), each of which is obtained by a simple translation scheme which replaces every multi-modal $\mathcal{PML}(\Diamond_s, \Diamond_r, \Diamond_u)$ formula φ by a standard $\mathcal{PML}$ formula φ^*. The translation $\varphi \mapsto \varphi^*$ leaves Boolean formulas unchanged and, otherwise, is defined as follows:

$$\begin{aligned}
(\Box_s\varphi)^* &= \Box\varphi^* & \\
([\mathtt{p}]_s\varphi)^* &= [\mathtt{p}]\varphi^* & \\
(\Box_a\varphi)^* &= \Box(a \to \Box\varphi^*) && \text{for } a \in \{r, u\} \\
([\mathtt{p}]_a\varphi)^* &= [\mathtt{p}](a \to [\mathtt{p}]\varphi^*) && \text{for } a \in \{r, u\}
\end{aligned}$$

The conjunction of the above formulas is satisfiable if and only if the corresponding instance of the tiling problem has a solution. The formula forces a grid-like structure, as depicted in Figure 5. Each point in the valuation of $0, 1, 2$ is assigned a tile type: the type of each such point i is required to match the type of its rightward successor $i + 2 \pmod 3$ and of its upward successor $i + 1 \pmod 3$. The rest of the argument is then analogous to the undecidability proof for $\mathcal{PML}(\Diamond_s, \Diamond_r, \Diamond_u)$. $\Box$

4 $\mathcal{PSL}$

Besides undecidability, another reason why one might be unhappy with $\mathcal{PML}$ as a language for modelling the poison game concerns the ⓟ operator. When used in conjunction with $\Diamond$, ⓟ allows to talk about Traveller's moves. However, ⓟ by itself does not have a counterpart in the poison game: neither player is allowed to roam the graph without poisoning, simply to check whether certain vertices are poisoned or not.

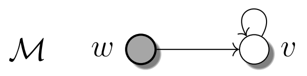
To overcome these difficulties, we will now consider the fragment of $\mathcal{PML}$ (and, *a fortiori*, $\mathcal{ML}_\emptyset$) that does not feature the basic $\Diamond$ modality, but which includes $\langle \mathtt{p} \rangle$ and the operator $\langle \mathtt{t} \rangle$, defined as $\langle \mathtt{t} \rangle\varphi \leftrightarrow \Diamond(\neg ⓟ \wedge \varphi)$. As before, the $\langle \mathtt{p} \rangle$ operator captures Poisoner's moves. The $\langle \mathtt{t} \rangle$ modality, on the other hand, captures Traveller's *safe* moves: i.e., moves along edges that do not lead to a poisoned state. We will refer to this logic as $\mathcal{PSL}$.

Expressive power. Even though there are no modalities that prompt the explicit deletion of edges, $\mathcal{PSL}$ is rather close to sabotage modal logic in spirit.[5] This is because we could also think of $\langle \mathtt{t} \rangle$ as the modality associated with a second graph relation, one that corresponds to Traveller's 'safe accessibility' relation and which shrinks over time, as more and more states get poisoned. Under this interpretation, the poison modality $\langle \mathtt{p} \rangle$ behaves analogously to the sabotage modality, in that poisoning moves result in the deletion of links from the safe accessibility relation, just as sabotaging moves trigger the

[5]This is why the logic is called $\mathcal{PSL}$, which is an abbreviation for *poison sabotage logic*.

deletion of links from the basic graph relation. The crucial difference between sabotage modal logic and $\mathcal{PSL}$, however, is that, while sabotaging only allows to remove one link at a time, poisoning prompts the deletion of *all* safe links leading to the poisoned state.

It is worth noting that $\langle\mathtt{p}\rangle$ and $\langle\mathtt{t}\rangle$ agree on atomic propositions and Boolean formulas, since here we restrict attention to the class of initial models where $P = \emptyset$. However, this is clearly not the case for arbitrary formulas. For a simple example, consider the formula $\langle\mathtt{p}\rangle\langle\mathtt{t}\rangle q \leftrightarrow \langle\mathtt{t}\rangle\langle\mathtt{t}\rangle q$, and the model $\mathcal{M} = (W, R, V, \emptyset)$ below, where $V(q) = \{v\}$. We then have that $\mathcal{M}, w \models \langle\mathtt{t}\rangle\langle\mathtt{t}\rangle q$, but $\mathcal{M}, w \not\models \langle\mathtt{p}\rangle\langle\mathtt{t}\rangle q$.



Like $\mathcal{ML}_\emptyset$ and $\mathcal{PML}$, $\mathcal{PSL}$ can express that Traveller has a strategy for surviving at least n rounds of a poison game via the inductive scheme below:

$$\rho_1 := [\mathtt{p}]([\mathtt{p}]\bot \vee \langle\mathtt{t}\rangle\top)$$

$$\rho_n := [\mathtt{p}]([\mathtt{p}]\bot \vee \langle\mathtt{t}\rangle\rho_{n-1})$$

We can also express the property that (i) the current state has a safely accessible successor that has itself as its only safely accessible successor, and that (ii) every state that is safely accessible from the current state is either an endpoint or has itself as its only safely accessible successor via the formula $\langle\mathtt{t}\rangle\langle\mathtt{t}\rangle\top \wedge [\mathtt{p}][\mathtt{t}]\bot$. It then follows that the tree model property fails for $\mathcal{PSL}$, too.

Now, note that the notion of $\mathcal{PML}$ bisimulation defined in Section 3 can be easily adapted to the case of $\mathcal{PSL}$ by replacing the standard clauses for $\Diamond$ with the following clauses for $\langle\mathtt{t}\rangle$:

Zig$_{\langle\mathtt{t}\rangle}$: if $(S, u)Z(S', u')$ and there exists $v \in W$ with $(u, v) \in R$ and $v \notin S$, then there exists $v' \in W'$ with $(u', v') \in R'$ and $v' \notin S'$, and $(S, v)Z(S', v')$;

Zag$_{\langle\mathtt{t}\rangle}$: if $(S, u)Z(S', u')$ and there exists $v' \in W'$ with $(u', v') \in R'$ and $v' \notin S'$, then there exists $v \in W$ with $(u, v) \in R$ and $v \notin S$, and $(S, v)Z(S', v')$.

Next, we make sure that the above definition is correct:

Proposition 4.1. *Let $\mathcal{M} = (W, R, V, P)$ and $\mathcal{N} = (W', R', V', P')$ be two $\mathcal{PSL}$ models, $w \in W$ and $w' \in W'$. If Z is a bisimulation linking (P, w) and (P', w'), then, for any $\varphi \in \mathcal{PSL}$,*

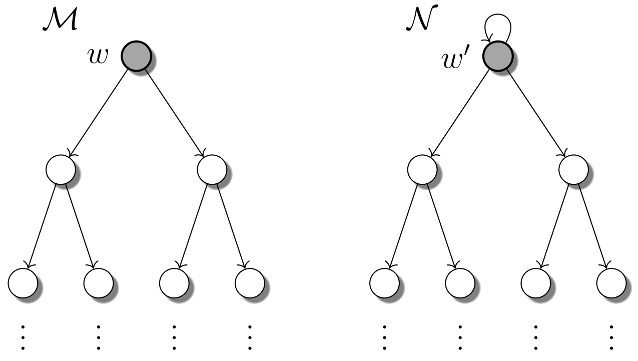
$$\mathcal{M}, w \models \varphi \textit{ iff } \mathcal{N}, w' \models \varphi.$$

Proof. The only case that requires checking is the one involving $\langle \mathtt{t} \rangle$. Suppose that $\mathcal{M}, w \models \langle \mathtt{t} \rangle \varphi$. Then, by the semantics of $\langle \mathtt{t} \rangle$, there is $v \in W$ with $(w, v) \in R$, $v \notin P$ and $\mathcal{M}, v \models \varphi$. By the $\mathbf{Zig}_{\langle \mathtt{t} \rangle}$ condition, we then have that there exists $v' \in W'$ with $(w', v') \in R'$, $v' \notin P'$ and $(P, v)Z(P', v')$. By the induction hypothesis, we can then conclude that $\mathcal{N}, v' \models \varphi$. Hence, $\mathcal{N}, w' \models \langle \mathtt{t} \rangle \varphi$. The other direction is analogous, except that it instead relies on the $\mathbf{Zag}_{\langle \mathtt{t} \rangle}$ condition. □

At this point, it is natural to ask how $\mathcal{PSL}$ compares with $\mathcal{PML}$ in terms of expressivity. Using the above notion of $\mathcal{PSL}$ bisimulation, we can show that $\mathcal{PSL}$ is strictly less expressive than $\mathcal{PML}$:

Proposition 4.2. *$\mathcal{PSL} < \mathcal{PML}$.*

Proof. Since $\mathcal{PSL}$ is a syntactic fragment of $\mathcal{PML}$, we trivially have that $\mathcal{PSL} \leq \mathcal{PML}$. To show that $\mathcal{PML} \not\leq \mathcal{PSL}$, consider the two infinite binary trees $\mathcal{M} = (W, R, V, \emptyset)$ and $\mathcal{N} = (W', R', V', \emptyset)$ shown below, where $V(q) = V'(q) = \emptyset$ for all proposition letters q.



The two pointed models $(\mathcal{M}, w)$ and $(\mathcal{N}, w')$ are $\mathcal{PSL}$ bisimilar. However, we have that $\mathcal{N}, w' \models_{\mathcal{PML}} \langle \mathtt{p} \rangle \Diamond \circled{\mathtt{p}}$, while $\mathcal{M}, w \not\models_{\mathcal{PML}} \langle \mathtt{p} \rangle \Diamond \circled{\mathtt{p}}$. □

Just as in the case of $\mathcal{PML}$, we can define a direct translation ST_y^X from $\mathcal{PSL}$ formulas to first-order formulas, where, once again, y is a variable and X a finite set of variables. We only give the clause for the $\langle \mathtt{t} \rangle$ operator:

$$\mathsf{ST}_y^X(\langle \mathtt{t} \rangle \varphi) = \exists z \Big(Ryz \wedge \bigwedge_{x \in X} \neg(z = x) \wedge \mathsf{ST}_z^X(\varphi) \Big)$$

Model checking. The model-checking complexity of $\mathcal{PSL}$ can be analysed in the exact same way as that of $\mathcal{PML}$. The standard translation into first-order logic (or simply the fact that $\mathcal{PSL}$ is a fragment of $\mathcal{ML}_\emptyset$) provides a PSPACE upper bound, while the lower bound can be shown via the same reduction from the true QBF problem. For the latter, it is sufficient to note that $\mathcal{PSL}$ can express survival for at least n rounds.

Theorem 4.3. *Model checking for $\mathcal{PSL}$ is* PSPACE*-complete.*

Proof. By the same reduction as in Theorem 3.5: given a QBF φ with n variables, construct the pointed model $(\mathcal{M}_\varphi, s)$. As we saw, $\mathcal{PSL}$ can express, through the formula ρ_n given above, the existence of a survival strategy for Traveller for at least n rounds. Then, Traveller has winning strategy in the poison game on $(\mathcal{M}_\varphi, s)$ if and only if $\mathcal{M}_\varphi, s \models \langle \mathtt{t} \rangle \rho_{n+3}$. □

The logic $\mathcal{PSL}$ appears to be particularly well-suited for talking about the poison game. It allows to describe each stage of the game from the local perspective of the vertex currently occupied by the players, and quantification is restricted so as to precisely match the possible moves of the players: the $\langle \mathtt{t} \rangle$ and $\langle \mathtt{p} \rangle$ modalities capture exactly statements of the form 'there is an available move by Traveller/Poisoner resulting in φ'. Note, in particular, the elegant way in which $\mathcal{PSL}$ expresses n-round survival for the two players: with every application of a modal operator corresponding to a player's move, the language allows to talk about the game steps with no frills. Thus, $\mathcal{PSL}$ seems to have a strong claim for being the most appropriate logic, at least among the memory logics discussed in this paper, for modelling the poison game from a local, stepwise perspective.

These considerations render the decidability question for $\mathcal{PSL}$ especially poignant. The question is open: should $\mathcal{PSL}$ satisfiability be decidable, we would then have a logic that strikes a very pleasing balance between close fit with the original game, expressivity and good computational behaviour.

5 Conclusion

In this paper, we studied three logics for modelling the poison game, moving from the memory logic $\mathcal{ML}_\emptyset$ to two fragments thereof: $\mathcal{PML}$ and $\mathcal{PSL}$. We showed that these logics form a chain in expressive power, with $\mathcal{PSL} < \mathcal{PML} < \mathcal{ML}_\emptyset$, and we introduced suitable notions of bisimulation for the two new logics presented in this paper. We proved that $\mathcal{PML}$, while strictly less expressive than $\mathcal{ML}_\emptyset$, has a PSPACE-complete model-checking problem and an undecidable satisfiability problem. We also showed that model checking for $\mathcal{PSL}$, a more sabotage-style logic for describing the poison game, is PSPACE-complete, and we concluded by identifying a natural open question: namely, whether the satisfiability problem for $\mathcal{PSL}$ is decidable.

Our results indicate that methods from modal logic are indeed well-suited for modelling graph games such as the poison game and the sabotage game—and, more broadly, 'evolving' relational structures. They also suggest the adoption of a useful technical perspective for this wider purpose: the systematic use of memory logics and, more generally, of techniques from hybrid logics (such as the spy-point method), in addition to game-reduction techniques for analysing model-checking complexity [Löding and Rohde, 2003a]. Lastly, our findings invite a broad methodological question concerning the emerging study of modal logics for graph games: which logics are the natural candidates for studying a given class of games, and how do such design choices affect significant properties of these logics?

Acknowledgments: We would like to thank Johan van Benthem for many helpful suggestions and inspiring discussions, the organisers and participants of the 4th Asian Workshop on Philosophical Logic (AWPL 2018) held at Tsinghua University, as well as the anonymous referees for their feedback and valuable comments.

References

C. Areces. Hybrid Logics: The Old and the New. In the *Proceedings of LogKCA-07,* San Sebastian, Spain, pages 15–29, 2007.

C. Areces, D. Figueira, S. Figueira, and S. Mera. The Expressive Power of Memory Logics. *Review of Symbolic Logic*, 4(2): 290–318, 2011.

C. Areces, R. Fervari, and G. Hoffmann. Relation-changing modal operators. *Logic Journal of the IGPL*, 23: 601–627, 2015.

G. Aucher, J. van Benthem, and D. Grossi. Sabotage Modal Logic: Some Model and Proof Theoretic Aspects. In the *Proceedings of the 5th International Workshop on Logic, Rationality and Interaction (LORI '15),* Taipei, Taiwan, pages 1–13, 2015.

G. Aucher, J. van Benthem, and D. Grossi. Modal Logics of Sabotage Revisited. *Journal of Logic and Computation*, 28(2): 269–303, 2017.

J. van Benthem. An Essay on Sabotage and Obstruction. In Hutter, D. and Stephan, W. (eds.), *Mechanizing Mathematical Reasoning*, Lecture Notes in Computer Science, 2605: 268–276, 2005.

J. van Benthem. *Logic in Games.* The MIT Press, Cambridge, MA, 2014.

J. van Benthem, K. Mierzewski, and F. Zaffora Blando. The Modal Logic of Stepwise Removal. Under review, 2019.

P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information,* Special issue on decompositions of first-order logic, 4(3): 251–272, 1995.

P. Duchet and H. Meyniel. Kernels in directed graph: a poison game. *Discrete Mathematics*, 115: 273–276, 1993.

A. Ehrenfeucht. An Application of Games to the Completeness Problem for Formalized Theories. *Fundamenta Mathematicae*, 49: 129–141, 1961.

R. Fraïssé. Sur quelques classifications des systèmes de relations. *Publications des Sciences de l'Université de l'Algérie, Série A1*, pages 35–182, 1954.

D. Grossi and S. Rey. Credulous Acceptability, Poison Games and Modal Logic. To appear in the *Proceedings of SYSMICS 2019*, 2019.

J. Hintikka. *Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic.* Clarendon Press, Oxford, 1973.

W. Hodges. *Building Models by Games.* Dover Publications, Mineola, NY, 2006.

W. Hodges. Logic and Games. *The Stanford Encyclopedia of Philosophy* (Spring 2013 Edition), Edward N. Zalta (ed.), 2013.

G. Hoffmann. Undecidability of a Very Simple Modal Logic with Binding. Preprint, 2015.

C. Löding and P. Rohde. Model Checking and Satisfiability for Sabotage Modal Logic. In Pandya, P. K. and Radhakrishnan, J. (eds.), *Proceedings of the 23rd Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2003),* Mumbai, India, December 15-17, 2003, Lecture Notes in Computer Science 2914, Springer-Verlag: 302–313, 2003a.

C. Löding and P. Rohde. Solving the Sabotage Game is PSPACE-hard. In Rovan, B. and Vojtás, P. (eds.), *Proceedings of the 28th International Symposium on the Mathematical Foundations of Computer Science (MFCS 2003),* Bratislava, Slovakia, August 25-29, 2003, Lecture Notes in Computer Science 2747, Springer-Verlag: 531–540, 2003b.

P. Lorenzen. *Einführung in die Operative Logik und Mathematik.* Springer-Verlag, Berlin, 1955.

S. Mera. Modal Memory Logics. Ph.D. Dissertation, Universidad de Buenos Aires, Buenos Aires, Argentina and Université Henri Poincaré, Nancy, France, 2009.

K. Mierzewski and F. Zaffora Blando. The Modal Logic(s) of Poison Games. Manuscript, Stanford University, 2016.

P. Rohde. On Games and Logics over Dynamically Changing Structures. Ph.D. Dissertation, RWTH Aachen University, Aachen, Germany, 2005.

M. Sipser. *Introduction to the Theory of Computation.* Third Edition, Cengage Learning, Boston, 2012.

G. Vreeswijk and H. Prakken. Credulous and Sceptical Argument Games for Preferred Semantics. In the *Proceedings of the 7th European Workshop on Logic for Artificial Intelligence (JELIA '00), LNAI*, pages 239–253, 2000.

T. Zhang. Solution Complexity of Local Variants of Sabotage Game. To appear in the *Proceedings of the Workshop on Logics for the Formation and Dynamics of Social Norms (LFDSN 2019),* Zheijang University, Hangzhou, China, May 4-5, 2019.