On bisimulation in modal logics for games on graphs: An algorithmic study

Sujata Ghosh[1], Shreyas Gupta[2], and Lei Li[3]⋆

[1] Indian Statistical Institute, Chennai, India
[2] Indian Institute of Science, Bengaluru, India
[3] School of Philosophy, Shaanxi Normal University, Xi'an, China.

Abstract.

We have been discussing modal operators modelling structural changes in graphs represented as moves in different graph games. For this work, the focus is on the corresponding notions of bisimulation that explore the similarities in these graph structures. We provide an algorithmic study of the same, for sabotage modal logic vis-à-vis the sabotage games in particular, paving the way for a general study to check whether different graph game models involving certain structural changes are bisimilar. This, in turn, provides us with the corresponding concepts of logical equivalences, especially for such model-changing modal logics. Through our algorithmic analyses we provide a PSPACE upper bound of the bisimulation / model comparison problem of sabotage modal logic and then generalise the result to similar other model-changing modal logics. We also provide some insight into the higher complexity of the model comparison problem for these logics compared to that for the basic modal logic.

Keywords: Modal logic, graph change, sabotage game, link deletion, algorithms.

1 Introduction

In two-player games on graphs we generally consider two players playing a turn-based game by moving a token through a directed graph, tracing out a finite or infinite path. Such games provide us with powerful tools to reason about various questions arising in diverse domains, e.g., computer science, logic, linguistics, economics, mathematics, philosophy, and biology. One can also consider different variants of such graph games where such variations can arise from different winning conditions (e.g., reachability, parity [16]), independent moves of players (e.g., cop and robber game [24]), one player obstructing moves of the others (e.g., sabotage game [29], poison game [12]) and others. In the interplay between game theory, logic and computer science, these graph games provide exploratory models for reactive systems that need to interact with the uncertain environment.

From the logic perspective, model-changing modal logics (e.g., see [9,8,35,31]) have played a significant role in describing strategic reasoning in games on graphs. With respect to relation changes in models, in particular, link/edge deletion in graphs, sabotage games [29] are natural examples where one player is concerned with a reachability objective and the other player is involved in obstructing her opponent's moves by deleting edges from the graph. Model-changing logics related to sabotage-style graph games with edge deletions are presented in [33,26,20,7,32]. With regard to domain changes in models, a game that is close to the spirit of describing point/vertex deletion on graphs is the poison game in [12]: one player is concerned with moving indefinitely in the game graph, and her opponent is involved in obstructing her moves by poisoning certain vertices whose effect is analogous to that of 'point deletion' from the perspective of the former player. To reason about poison games, model-changing logics *PSL* and *PML* in [36,17] use operators for changing valuations in and/or domains of models, which are inspired by memory logics [36,23,4]. In addition, operators involving such changing of valuations are also mentioned in [28,33] and, point-deletion style operators have been proposed in [29,13,34,2].

The logics that we are talking about aim to capture three mechanisms of model transformation, namely, those describing domain-changes, relation-changes and valuations-changes in models and their combinations. As such, we concentrate on various extensions of basic modal logic with the new operator $\langle up \rangle$, which we call $MCML(up)$, where $\langle up \rangle$ reflects various mechanisms of model transformation. In particular, we deal with the bisimulation / model comparison problems of such model-changing logics. In model-theoretic studies of modal logics, the notion

⋆ Corresponding author, e-mail:lileity@snnu.edu.cn

of bisimulation plays a central role as formally, whenever a mathematical structure or a model is introduced, a notion of invariance of such models comes up by default. And, bisimulation is the notion of model-invariance with respect to modal languages. Thus it comes up quite naturally in a study featuring the landscape of model changing modal logics. In addition, from the viewpoint of the logics of game boards, this notion of bisimulation paves the way for measuring equivalence of 'game boards' that is, the graphs, for the purposes of playing the game. For example, a question to ponder upon while considering such notions of game equivalence: What simplest graph in an equivalence class still shows the essential structure of the game? Moreover, from the context of the underlying graph structure and the dynamic nature of our study, one could also think studying novel invariance notions of graph dynamics, based on the different bisimulation notions described below. Last but not the least, across all these viewpoints, the notion of bisimulation or model comparison aids in the study of the expressive power of the corresponding modal languages.

In this work, we provide an algorithmic study of the model comparison problem in sabotage modal logic to instantiate a particular case of the uniform algorithmic study of the said problem for different model-changing modal logics, as discussed in [15]. The main idea is to shed some light on the complexity of the problem. But, before focusing on the particular study, we provide a general introduction to the different logics. We consider the operators $\langle sb \rangle$ and $\langle gsb \rangle$ [2,5,13,32,26] to model edge deletion in models, $\langle br \rangle$ and $\langle gbr \rangle$ [13] to model edge addition in models, and $\langle sw \rangle$ and $\langle gsw \rangle$ [3,2,13] to model arrow swap in models. In addition, we consider $\langle de \rangle$ [29] and $\langle ch \rangle$ [28] for point deletion and valuation change in models, respectively. Such a study provides insight into the complexity of the bisimulation problems of these modal logics which have not been studied before.

We note here that there is a strand of literature exploring technical properties of these logics. $MCML(gsb)$ was first introduced in [29], and complete proof systems for $MCML(gsb)$ have been discussed in [5,13]. For the decidability and complexity questions, we have the following results [22,13,21,28]: (i) for $\langle up \rangle \in \{\langle sb \rangle, \langle gsb \rangle, \langle sw \rangle, \langle br \rangle, \langle ch \rangle\}$, the satisfiability problem for $MCML(up)$ is *undecidable*, and (ii) for $\langle up \rangle \in \{\langle sb \rangle, \langle gsb \rangle, \langle sw \rangle, \langle gsw \rangle, \langle br \rangle, \langle gbr \rangle\}$, the model-checking problem for $MCML(up)$ is *PSPACE-complete*. The complexity of bisimulation or the model comparison problem has been explored in [15], and we provide a brief survey of the results with a special focus on sabotage modal logic. From a game-theoretic perspective, the notion of bisimulation not only helps us to understand the expressive power of logics on game boards, but also measures the equivalence of games (graphs) for the purposes of playing the game. We should mention here that the model comparison problem has been in the radar of researchers for a long time. In addition to the development of verification algorithms [11,14,10], model comparison problem also holds fundamental importance in the field of concurrency theory and related areas of computer science [1,18]. It is well-known that deciding bisimilarity over finite labelled transition systems is in deterministic polynomial time [25,6,19]. To the best of our knowledge, complexity study of the bisimulation problems concerning these model-changing logics is still open. Solving this problem will, on one hand, provide us with a finer understanding of the practical applicabilities of these logics, and on the other hand, provide us with better insights about their expressive powers. In this work, we provide *PSPACE* upper bounds for the bisimulation problems for all the model-changing modal logics described above. Our quest for the lower bounds for these problems did not deliver any result at the current point, and we leave '*finding lower bounds for these bisimulation problems*' as open questions.

The rest of this work can be summarized as follows: In the following section, we introduce the relevant logic frameworks together with their respective notions of bisimulations. The next one gives us a detailed algorithmic study together with upper bounds of the complexity of the relevant problems. The final section provides some further related results and concludes the paper with a discussion on the lower bound.

2 A general framework

Following [15], we describe the various model-changing logics that we are going to base our study on. We also recapitulate the corresponding notions of bisimulation. The main focus will be on the logics describing relation updates where the domains remain fixed. To have a uniform description of these logics, we start with a general framework followed by the specific ones.

2.1 A uniform language

Given a countable, infinite set of propositional variables Prop, the syntax of the general model-changing modal logic $MCML(up)$ is given as follows:

$\varphi : p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \langle up\rangle\psi$,

where $p \in \mathsf{Prop}$, $\langle up\rangle$ is a model-update modality. The dual $[up]\psi$ formula is defined as usual: $\neg\langle up\rangle\neg\psi$.

The models for $MCML(up)$ are given by usual relational models $\mathcal{M} = (W, R, V)$ for modal logics, where, W is a non-empty set, $R \subseteq W \times W$, and $V : W \to 2^{\mathsf{Prop}}$. A pair $(\mathcal{M}, w)$, where $w \in W$ is called a *pointed model*. Let $\mathfrak{M}$ denote the class of all pointed models, and r_{up} be a subset of $\mathfrak{M} \times \mathfrak{M}$ corresponding to the operator $\langle up\rangle$. Given a pointed model $(\mathcal{M}, w)$, the set $\{(\mathcal{M}', w') \mid ((\mathcal{M}, w), (\mathcal{M}', w')) \in r_{up}\}$ collects all the updated pointed models from $(\mathcal{M}, w)$ that we get with respect to the operator $\langle up\rangle$. The truth definition of the formulas of $MCML(up)$ in pointed models are as usual for the boolean and the modal formulas, and for the operator $\langle up\rangle$, it is given as follows:

- $(\mathcal{M}, w) \models \langle up\rangle\psi$ *iff there is a pointed model* $(\mathcal{M}', w')$ *with* $((\mathcal{M}, w), (\mathcal{M}', w')) \in r_{up}$ *and* $(\mathcal{M}', w') \models \psi$.

With the syntax and semantics out of the way, we now focus on the following question which forms the backbone of this work: when do two pointed models satisfy the same formulas under the language $MCML(up)$? The definition of the relevant bisimulation concept, that is, $\langle up\rangle$-bisimulation is given as follows.

Let $\mathcal{M}_1 = (W_1, R_1, V_1)$ and $\mathcal{M}_2 = (W_2, R_2, V_2)$ be two relational models. A non-empty relation Z over a set of pointed models is an $\langle up\rangle$-bisimulation between $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$, denoted by $(\mathcal{M}_1, w_1)Z(\mathcal{M}_2, w_2)$, if the following conditions are satisfied:

(1). **Atom**: If $(\mathcal{M}_1, w_1)Z(\mathcal{M}_2, w_2)$, then $(\mathcal{M}_1, w_1) \models p$ iff $(\mathcal{M}_2, w_2) \models p$ for all atomic propositions $p \in \mathsf{Prop}$.
(2). **Zig**$_\Diamond$: If $(\mathcal{M}_1, w_1)Z(\mathcal{M}_2, w_2)$, and there exists $v_1 \in W_1$ such that $w_1R_1v_1$, then there is a $v_2 \in W_2$ such that $w_2R_2v_2$ and $(\mathcal{M}_1, v_1)Z(\mathcal{M}_2, v_2)$.
(3). **Zag**$_\Diamond$: Same as above in the converse direction.
(4). **Zig**$_{up}$: If $(\mathcal{M}_1, w_1)Z(\mathcal{M}_2, w_2)$, and there exists a pointed model $(\mathcal{M}'_1, u_1)$ such that $((\mathcal{M}_1, w_1), (\mathcal{M}'_1, u_1)) \in r_{up}$, then there exists a pointed model $(\mathcal{M}'_2, u_2)$ such that $((\mathcal{M}_2, w_2), (\mathcal{M}'_2, u_2)) \in r_{up}$ and $(\mathcal{M}'_1, u_1)Z(\mathcal{M}'_2, u_2)$.
(5). **Zag**$_{up}$: Same as above in the converse direction.

Note that the definition above is given in a generalized way, we shall make changes below according to the specific operators. Generally speaking, there are three cases.

– We do not need to make any adjustments, the definition may fit well for the operator $\langle up\rangle$ under consideration.
– The dynamics of the models, that the operator $\langle up\rangle$ reflects, may be quite complicated. Then, an abundant amount of information may be wrapped up in the respective definitions of r_{up} that we shall process further with respect to the items (4) and (5). For example, in the item (4), '$((\mathcal{M}_1, w_1), (\mathcal{M}'_1, w'_1)) \in r_{up}$' may involve complex formulas being satisfied at certain points, which shall be translated into additional conditions for bisimulation. In such cases, we shall restate the items (4) and (5) in the terms of the specific forms of the operator $\langle up\rangle$.
– Alternatively, the operator $\langle up\rangle$ may not increase the expressivity of the logic, which means that for any formula with $\langle up\rangle$, there is an equivalent formula without it. In such cases, items (4) and (5) become redundant, and we shall not consider them.

Thus, we treat the definition of bisimulation above in a broader perspective and many specific instances will be taken up later where we will delve into the minute details. Based on this definition, we can prove that bisimulation implies modal equivalence which we claim formally in the following. For simplicity, if there is an $\langle up\rangle$-bisimulation between two pointed models $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$, we call them $\langle up\rangle$-bisimilar.

Proposition 1. *If two pointed models* $(\mathcal{M}_1, w_1)$ *and* $(\mathcal{M}_2, w_2)$ *are* $\langle up\rangle$*-bisimilar, then they satisfy the same formulas of the logic* $MCML(up)$.

Proof. We can prove this by applying induction on the structure of formulas, and we only focus on the formula of the form $\langle up\rangle\psi$. Suppose that $(\mathcal{M}_1, w_1) \models \langle up\rangle\psi$. Then there is $(\mathcal{M}'_1, u_1)$ such that $((\mathcal{M}_1, w_1), (\mathcal{M}'_1, u_1)) \in r_{up}$, and $(\mathcal{M}'_1, u_1) \models \psi$. According to the definition of $\langle up\rangle$-bisimulation, there exists $(\mathcal{M}'_2, u_2)$ such that $((\mathcal{M}_2, w_2), (\mathcal{M}'_2, u_2)) \in r_{up}$ and $(\mathcal{M}'_1, u_1)Z(\mathcal{M}'_2, u_2)$. we have $(\mathcal{M}'_2, u_2) \models \psi$ by I.H., it follows that $(\mathcal{M}_2, w_2) \models \langle up\rangle\psi$.

2.2 On specific ones

We have proposed the language $MCML(up)$ for describing certain model-changing logics in a uniform way and the corresponding notion of bisimulation. Next we will demonstrate the specific notions of bisimulations with respect to the specific logics.

A number of model-changing operators have been proposed over the years which are basically modelling different dynamic mechanisms. We now investigate some of these modal operators characterizing basic mechanisms of model-changing. The operators $\langle sb\rangle$, $\langle gsb\rangle$, $\langle sw\rangle$, $\langle gsw\rangle$, $\langle br\rangle$ and $\langle gbr\rangle$ are proposed to capture relation-changing in models, while $\langle de\rangle$ is proposed to characterize domain-changing in models (followed by relation-changes), and $\langle ch\rangle$ for valuation-changing. We have chosen these operators as representatives for expressing the three different kinds of model-changing operations: (i) domain-changing, (ii) relation-changing (with domain remaining fixed) and (iii) valuation-changing (with domain and relation remaining fixed). The intuitive meaning of these operators are as follows.

- $\langle sb\rangle\psi$ can be read as 'it is the case that ψ, after we sabotage some arrow starting at the present point'.
- $\langle gsb\rangle\psi$ can be read as 'it is the case that ψ, after we sabotage some arrow in the model'.
- $\langle sw\rangle\psi$ can be read as 'it is the case that ψ, after we swap some arrow starting at the present point'.
- $\langle gsw\rangle\psi$ can be read as 'it is the case that ψ, after we swap some arrow in the model'.
- $\langle br\rangle\psi$ can be read as 'it is the case that ψ, after we add a new arrow at the present point'.
- $\langle gbr\rangle\psi$ can be read as 'it is the case that ψ, after we add a new arrow in the model'.
- $\langle de\rangle\psi$ can be read as 'it is the case that ψ, after some point is deleted from the model'.
- $\langle ch\rangle\psi$ can be read as 'it is the case that ψ, after the valuation at the present point is updated'.

All these operators have been studied extensively in the literature. The operators $\langle sb\rangle$, $\langle br\rangle$ appear in [2,13], $\langle gsb\rangle$ appears in [2,5,13,32,26], $\langle sw\rangle$ appears in [3,2,13], $\langle gsw\rangle$, $\langle gbr\rangle$ are proposed in [13], $\langle de\rangle$ occurs in [29] and $\langle ch\rangle$ is proposed in [28] (with $\bigcirc$ expressing the same). We now define the corresponding r_{up}s'.

Let $\mathcal{M}_1 = (W_1, R_1, V_1)$ and $\mathcal{M}_2 = (W_2, R_2, V_2)$ be two models with $w \in W_1, v \in W_2$. We give the specific definitions of r_{up}, where $\langle up\rangle$ can be the operators we mentioned above. We have that $((\mathcal{M}_1, w), (\mathcal{M}_2, v)) \in r_{up}$ if the following holds:

- $\langle sb\rangle$: $W_2 = W_1$, $(w, v) \in R_1$, $R_2 = R_1\backslash\{(w, v)\}$ and $V_2 = V_1$.
- $\langle gsb\rangle$: $W_2 = W_1, R_2 = R_1\backslash\{(w_1, w_2)\}$ for some $(w_1, w_2) \in R_1$, $V_2 = V_1$ and $w = v$.
- $\langle sw\rangle$: $W_2 = W_1$, $(w, v) \in R_1$,$R_2 = R_1\backslash\{(w, v)\} \cup \{(v, w)\}$ and $V_2 = V_1$.
- $\langle gsw\rangle$: $W_2 = W_1, R_2 = R_1\backslash\{(w_1, w_2)\} \cup \{(w_2, w_1)\}$ for some $(w_1, w_2) \in R_1$, $V_2 = V_1$ and $w = v$.
- $\langle br\rangle$: $W_2 = W_1$, $(w, v) \notin R_1$, $R_2 = R_1 \cup \{(w, v)\}$ and $V_2 = V_1$.
- $\langle gbr\rangle$: $W_2 = W_1, R_2 = R_1 \cup \{(w_1, w_2)\}$ for some $(w_1, w_2) \notin R_1$, $V_2 = V_1$ and $w = v$.
- $\langle de\rangle$: $W_2 = W_1\backslash\{w_1\}$ for some $w_1 \neq w$ in W_1, $R_2 = \{(u, v) \in R_1 \mid u \neq w_1 \; and \; v \neq w_1\}$, $V_2(u) = V_1(u)$ for all $u \in W_2$ and $w = v$.
- $\langle ch\rangle$: $W_2 = W_1, R_2 = R_1$, $V_2(w) = A$ and $V_2(u) = V_1(u)$ for $u \neq w$, where A is a set of proposition letters, and $w = v$.

Intuitively, the truth conditions of the above operators can be displayed in Figure1–8. For example, in Figure1, $\langle sb\rangle\varphi$ is true at $(\mathcal{M}_1, w)$, if and only if there exists pointed model $(\mathcal{M}_2, v)$ with $((\mathcal{M}_1, w), (\mathcal{M}_2, v)) \in r_{sb}$ such that φ is true at $(\mathcal{M}_2, v)$. It is worth mentioning that when $\langle up\rangle$ is $\langle ch\rangle$, we have a single item to replace the items (4) and (5) as follows.

($\star$) $(W_1, R_1, V_1, w_1)Z(W_2, R_2, V_2, w_2)$ implies $(W_1, R_1, {V_1}_A^{w_1}, w_1)Z(W_2, R_2, {V_2}_A^{w_2}, w_2)$ for every $A \subseteq \mathsf{Prop}$, where for $i = 1, 2$, ${V_i}_A^{w_i}$ is almost V_i, except ${V_i}_A^{w_i} = A$.

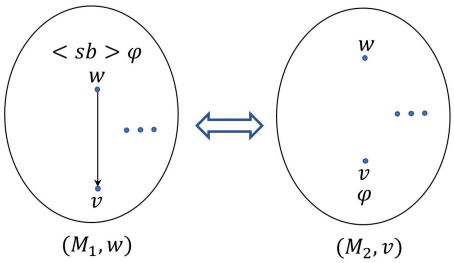

FIG. 1: $\langle sb \rangle \varphi$

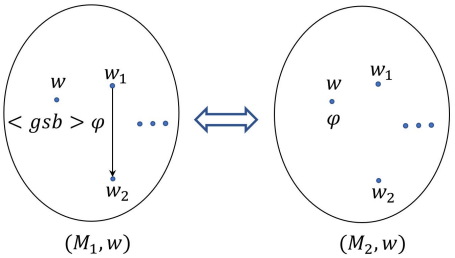

FIG. 2: $\langle gsb \rangle \varphi$

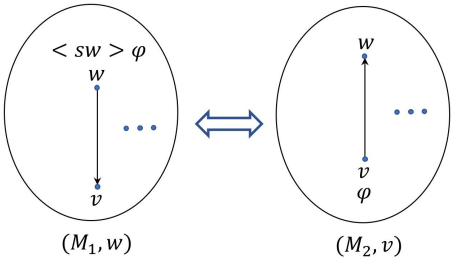

FIG. 3: $\langle sw \rangle \varphi$

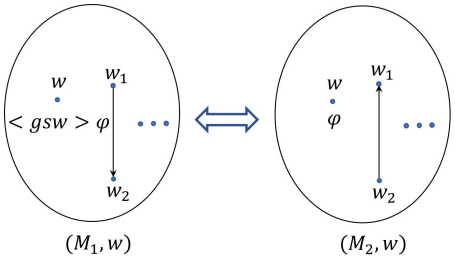

FIG. 4: $\langle gsw \rangle \varphi$

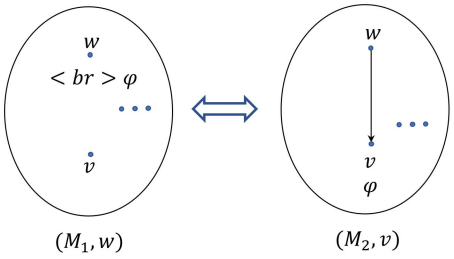

FIG. 5: $\langle br \rangle \varphi$

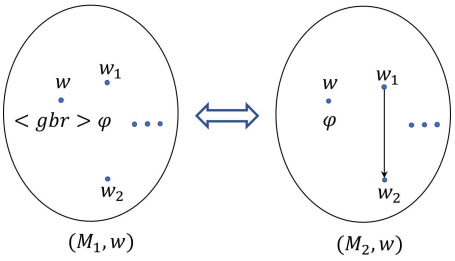

FIG. 6: $\langle gbr \rangle \varphi$

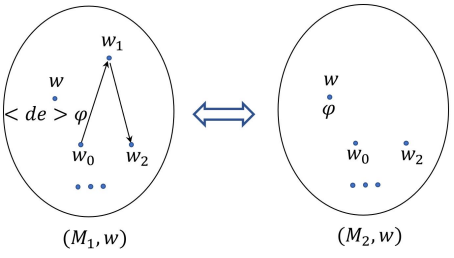

FIG. 7: $\langle de \rangle \varphi$

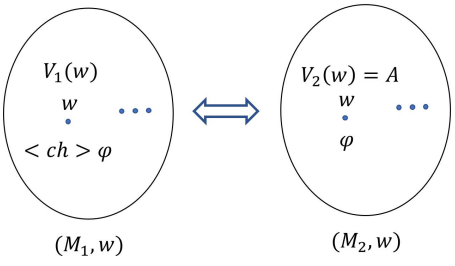

FIG. 8: $\langle ch \rangle \varphi$

Meanwhile, Proposition 1 still works when we unfold the definitions of $\langle up \rangle$-bisimulation for different operators.

For basic modal logic, we can use the notion of bisimulation to reduce equivalently a given model to a smaller one. Moreover, given a finite model M, we can find a minimal model and it is bisimilar to the original model M, i.e., the bisimulation contraction [30]. With these distinct notions of $\langle up \rangle$-bisimulation above, do we have the corresponding notions of $\langle up \rangle$-bisimulation contraction? For finite models, the simple answer is Yes.

Fix a finite pointed model (M_1, w_1), we aim to find a minimal model M_2, and a world w_2, such that (M_1, w_1) is $\langle up \rangle$-bisimilar to (M_2, w_2), there is a straightforward method to obtain such a result. The approach is to systematically test all models that are smaller than M_1 and check if they are $\langle up \rangle$-bisimilar. The key to this method is to have an algorithm that can determine whether two pointed models are $\langle up \rangle$-bisimilar, and it is desirable that the algorithm is efficient. Hence, we will investigate the complexity of the following decision problem: given two relational models, are they $\langle up \rangle$-bisimilar?

3 An algorithmic study

A general algorithmic study for the model comparison problem of various model-changing modal logic has been provided in [15]. In the following, we focus on sabotage modal logic, denoted as $\langle gsb \rangle$ above. Here, we provide an algorithm to check whether two pointed models are $\langle gsb \rangle$-bisimilar, or sabotage bisimilar (s-bisimilar for short). A very similar algorithm can be given to check bisimilarity in any of the models provided above.

3.1 The algorithm(s)

We define a function s-Bisimilar (Algorithm 1) that takes as input two pointed relational models, $(\mathcal{M}_1, w_1)$, $(\mathcal{M}_2, w_2)$ and a list $L \subseteq W_1 \times W_2$, where $\mathcal{M}_1 = (W_1, R_1, V_1)$ and $\mathcal{M}_2 = (W_2, R_2, V_2)$, and outputs "Yes" if the two models are s-bisimlar and the function is called with $L = \emptyset$, and "No" when the given models are not s-bisimilar and the function is called with $L = \emptyset$. We will argue its correctness and complexity following the algorithm.

Corresponding generalized algorithm is presented in [15]. With this algorithm out of the way, we are now ready to show the correctness of these algorithms. We will show a couple of cases in all details. The rest follow similarly.

3.2 Correctness

We now give a proof sketch for the correctness of algorithm 1. A detailed proof is presented in [15].

Theorem 1. *Given two models $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$, where $\mathcal{M}_1 = (W_1, R_1, V_1)$, $\mathcal{M}_2 = (W_2, R_2, V_2)$, $w_1 \in W_1$ and $w_2 \in W_2$; $(\mathcal{M}_1, w_1) \underline{\leftrightarrow}_s (\mathcal{M}_2, w_2)$ iff the function s-Bisimilar$((\mathcal{M}_1, w_1), (\mathcal{M}_2, w_2), \emptyset)$ returns yes. Here, by $(\mathcal{M}_1, w_1) \underline{\leftrightarrow}_s (\mathcal{M}_2, w_2)$ we will denote that $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$ are $\langle gsb \rangle$-bisimilar.*

Proof sketch: Sabotage bisimilarity checks if five conditions are satisfied. The algorithm follows similarly to check these five conditions. One of the main problems that comes in implementation is when the given models have cycles. As we have to check the s-bisimilarity for the successors too, the process may not terminate in case the given model is pointed at a node that is part of a cycle. We take care of this problem by maintaining a list of edges already travelled. We initialize the algorithm with this list being empty, and keep adding edges that we have travelled before changing the models. We again make the list empty after the model changing step. This is the only difference between the algorithm and the definition and hence correctness of the algorithm follows easily. □

Another way to think about writing this algorithm might be to use the existing algorithm for modal bisimulation (which is a poly-time algorithm) and add on to it to take care of the extra conditions. This type of approach does not directly work as it is not enough to check the satisfaction of the extra conditions for the two bisimliar models. Take the following example (cf. Figure 9). The models $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$ are bisimilar in the basic modal logic sense. Moreover, the pointed models $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$ also satisfy the (4) and (5) condition of the definition of $\langle gsb \rangle$-bisimilarity. But these models are not $\langle gsb \rangle$-bisimilar. To see this, assume on the contrary that $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$ are indeed $\langle gsb \rangle$-bisimilar. Then, $(\mathcal{M}_1, u_1)$ and $(\mathcal{M}_2, u_2)$ must be $\langle gsb \rangle$-bisimilar as well. But if we delete e_1 from $\mathcal{M}_1$, there is no edge in $\mathcal{M}_2$ such that $(\mathcal{M}_1, u_1)$ and $(\mathcal{M}_2, u_2)$ are even bisimilar.

Algorithm 1: Algorithm to check whether two pointed models are s-bisimilar

Input: $((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$
Initialize:L=$\emptyset$
Function s-Bisimilar($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2), L$):
  if $|R_1| \neq |R_2|$ **then**
    return No;
  forall *atomic propositions p* **do**
    if $(((w_1 \in V_1(p)) AND (w_2 \notin V_2(p)))\ OR\ ((w_1 \notin V_1(p)) AND (w_2 \in V_2(p))))$ **then**
      return No;
  forall $e_1 \in R_1$ **do**
    Found=0;
    forall $e_2 \in R_2$ **do**
      if *s-Bisimilar*$(((W_1, R_1 \backslash \{e_1\}, V_1), w_1), ((W_2, R_2 \backslash \{e_2\}, V_2), w_2), \emptyset)$==*Yes* **then**
        Increment Found;
        break;
    if *Found =0* **then**
      return No;
  forall $e_2 \in R_2$ **do**
    Found=0;
    forall $e_1 \in R_1$ **do**
      if *s-Bisimilar*$(((W_1, R_1 \backslash \{e_1\}, V_1), w_1), ((W_2, R_2 \backslash \{e_2\}, V_2), w_2), \emptyset)$==*Yes* **then**
        Increment Found;
        break;
    if *Found =0* **then**
      return No;
  if $(w_1, w_2) \notin L$ **then**
    forall $u_1 \in W_1$ **do**
      Found=0;
      forall $u_2 \in W_2$ **do**
        if $((w_1 R_1 u_1)\ AND\ (w_2 R_2 u_2))$ **then**
          if $((u_1, u_2) \notin L)$ **then**
            if *s-Bisimilar*$(((W_1, R_1, V_1), u_1), ((W_2, R_2, V_2), u_2), L \cup \{(w_1, w_2)\})$==*Yes* **then**
              Increment Found;
          else
            Increment Found
      if *(Found=0) AND* $(w_1 R_1 u_1)$ **then**
        return No;
    forall $u_2 \in W_2$ **do**
      Found=0;
      forall $u_1 \in W_1$ **do**
        if $((w_1 R_1 u_1)\ AND\ (w_2 R_2 u_2))$ **then**
          if $((u_1, u_2) \notin L)$ **then**
            if *s-Bisimilar*$(((W_1, R_1, V_1), u_1), ((W_2, R_2, V_2), u_2), L \cup \{(w_1, w_2)\})$==*Yes* **then**
              Increment Found;
          else
            Increment Found
      if *(Found=0) AND* $(w_2 R_2 u_2)$ **then**
        return No;
  return Yes;

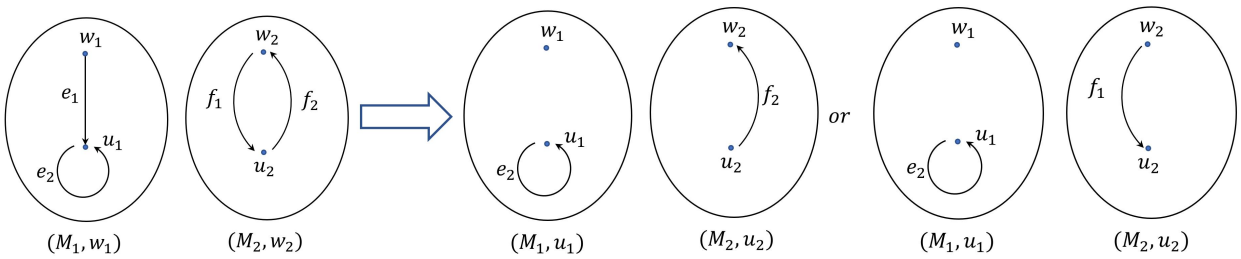

FIG. 9: Counterexample for $\langle gsb \rangle$-bisimilar

An example In what follows, we present an example run of the algorithm for s-Bisimilar. Proposition p is true in all the worlds of both models. Figure 10 shows all the important nodes and recursive calls.

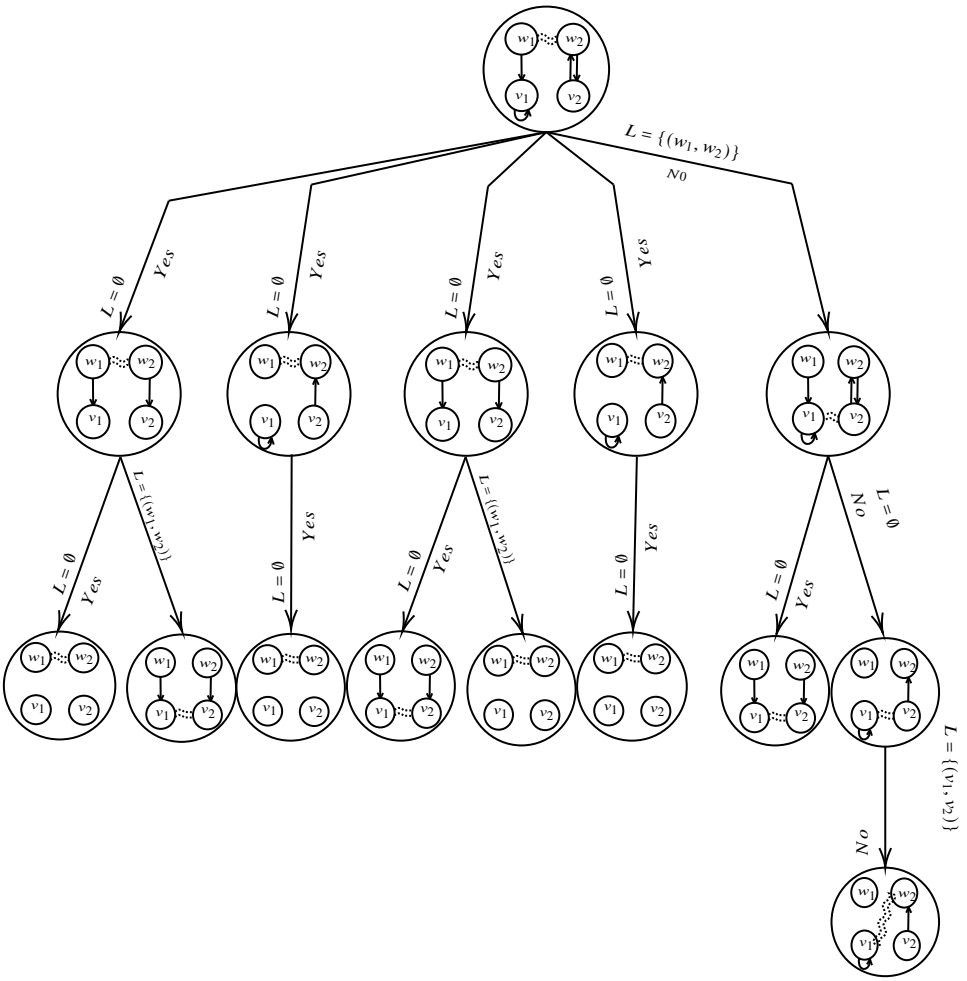

FIG. 10: Recursive graph

In the above run of the algorithm, not all children of the root return 'Yes', and hence the input models are not $\langle gsb \rangle$-*bisimilar*.

On complexity We now show that the complexity of the $\langle gsb \rangle$-bisimulation problem is in *PSPACE*.

Theorem 2. *Algorithm 1 terminates and is in* ***PSPACE****.*

Proof. We will form a recursion tree in Figure 11 to see whether the algorithm terminates and analyze the space complexity of the function.

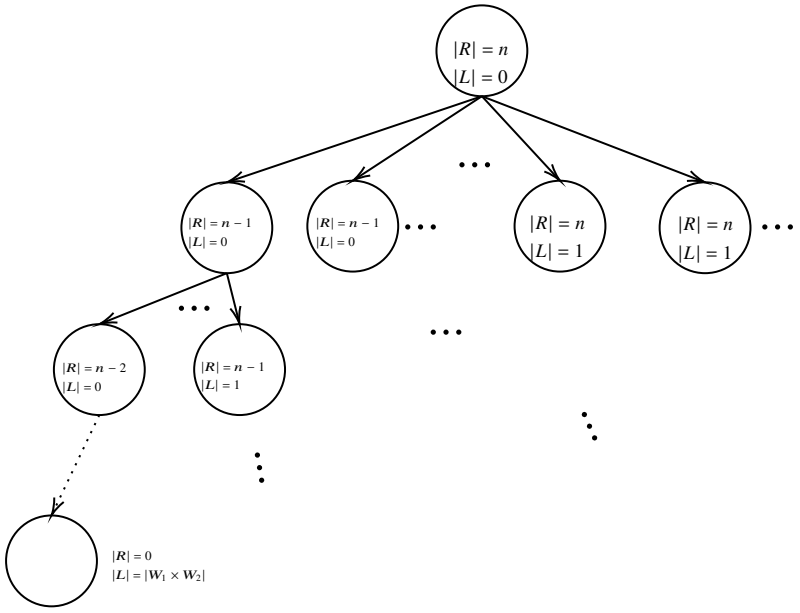

FIG. 11: A recursive tree

- When the input models have different number of edges, the algorithm terminates without any recursion. The algorithm takes the space required in one instance of the function. The function defines constant number of variables that need to be accounted for in terms of space in addition to the input. So, an instance of the function takes $O(1)$ space.
- When the two models have same number of edges, algorithm is called recursively. The number of edges in the models for each successive call is strictly less than n (namely, $n-1$). Another thing to note is that $|L|$ strictly increases in each successive call. Next it should be noted that the function call is not made if $|L| = |W_1 \times W_2|$. With these observations, we can bound the depth of recursion tree by $|R_1| \times |W_1 \times W_2|$. This shows that the algorithm terminates.

With the above observations, we see that the depth of the recursion tree is bounded by $|R_1| \times |W_1 \times W_2|$. Therefore, the space used by the algorithm is $s \times |R_1| \times |W_1 \times W_2|$, where s is the space used by one instance of the algorithm. The algorithm defines constant number of variables, which take space other than the input. So, once again, one instance of the function takes $O(1)$ space. Therefore, space used by whole run of Algorithm 1 is $|R_1| \times |W_1 \times W_2|$ which is a polynomial function in the size of the input.

3.3 Bisimulation for other model changing logics

We provided an algorithm for sabotage bisimulation. What about the algorithms for the other notions of bisimulation? We note here that the only differences in the definitions of the distinct notions of bisimulation are in conditions (4) and (5). We also observe that the function s-Bisimilar can be viewed to comprise of 5 parts corresponding to the 5 conditions of bisimulation. For Algorithm 1, the line numbers 7-12 and 13-18 correspond to the checks for the conditions (4) and (5), respectively. The corresponding algorithm for other logics are similar in nature. Therefore the correctness

proofs for other notions of bisimilarity are very similar to that of the case of s-bisimilarity. Thus, the complexity for all bisimilarity problems remains to be in **PSPACE**. A detailed study, along with the generalized algorithm, is provided in [15] - the main result is as follows:

Theorem 3. *Given two pointed relational models,* $(\mathcal{M}_1, w_1)$ *and* $(\mathcal{M}_2, w_2)$*, the problem of deciding whether they are* $\langle up \rangle$*-bisimilar is in* ***PSPACE*** *when* $\langle up \rangle \in \{\langle sb \rangle, \langle gsb \rangle, \langle sw \rangle, \langle gsw \rangle, \langle br \rangle, \langle gbr \rangle, \langle de \rangle, \langle ch \rangle\}$.

4 Further remarks

In this work, we have presented several existing modal logics concerned with the model changing phenomena and studied the notion of model comparison or bisimulation for one of these logics, viz. sabotage modal logic, from the algorithmic point of view. In the process, we have also shed some light on the complexity of the problem. As mentioned above, we have only been able to show upper bound result, and not any lower bound result. We now provide some discussions on the lower bounds of these complexity problems, not only for sabotage modal logic, but similar other model changing modal logics (cf. Section 2). We note that the results on tight bounds will complete this whole study.

Before going there, let us first note that the complexity for checking whether given two pointed models are bisimilar, in basic modal logic, is known to be in polynomial time [25]. What exactly makes the problem of $\langle up \rangle$-bisimilarity more complex (strictly more complex if **PTIME** is different than **PSPACE**)? The additional conditions (4) and (5) in the definition of $\langle up \rangle$-bisimilarity, compared to that of basic modal logic bisimilarity, requires a function that assigns a sequence of model-changing actions corresponding to one model to a sequence of model-changing actions in the other model. Formally, it requires a bijection $f : N(C_1) \to N(C_2)$ with $N(C)$ denoting the set of sequences of actions corresponding to the model-changing operator C. The function f should additionally satisfy the condition that any sequence of length n is mapped to a sequence of length n, for every n in $\mathbb{N}$. If $|C_1| = |C_2| = m$, then there are $2^{m^{2^m}}$ such functions. Given such a function, we need to check whether it satisfies the corresponding conditions for $\langle up \rangle$-bisimilarity on top of the models being bisimilar in the sense of basic modal logic. These conditions are what make this problem of $\langle up \rangle$-bisimilarity more complex. If we can show that every such function that satisfies the conditions for $\langle up \rangle$-bisimilarity is generated by a function $g : C_1 \to C_2$, then we believe that the complexity of $\langle up \rangle$-bisimilarity drops to the class **NP**. To draw an analogy, deciding whether given two graphs are isomorphic is in **NP**, but finding the isomorphism mapping may be more complex. This is equivalent to saying that given a small (with a number of elements bounded by a polynomial in the size of the input models) candidate generator of the relation $\langle up \rangle$-bisimilar, it may be efficient to check whether such a candidate can be extended to a full $\langle up \rangle$-bisimilar relation.

With regard to lower bound results, a general way one shows that a problem is C-hard, for a complexity class C, is by providing a polynomial-time reduction from an already known C-complete problem to the problem in question. The problem for determining whether a fully quantified Boolean formula is true or false, that is, the *TQBF* problem [27], is known to be **PSPACE**-complete, and has been used to prove **PSPACE**-hardness of many logic-related decision problems. However, finding such polynomial time reductions from the *TQBF* and similar other **PSPACE**-complete problems to the model comparison problem for sabotage logic have turned out to be quite involved. Complications arose even for showing **NP**-hardness results for these model comparison problems using **NP**-complete *SAT* and other problems. There are some problems for which such a reduction might be easy, for example, graph isomorphism problem. But again, such a reduction would not be helpful as the problem of graph isomorphism itself does not have a known lower bound.

Another way to get a tight lower bound is to follow an analogous strategy as done to show **PSPACE** hardness of *TQBF*. In this approach one proves that for any language $\mathcal{L}$ with an instance x, $M(x) = 1$ iff $\Psi \in TQBF$, where $M()$ is a PSPACE Turing machine that decides $\mathcal{L}$. To follow this approach, we first need to choose this complexity class, say C, and then reduce every problem instance of an arbitrary language to an instance of the problem at hand. This approach seems to work for *TQBF* especially due to the structure of the problem but somewhat hard to replicate in an altogether different kind of a problem.

Yet another natural approach is to analyse a sub-problem of the current problem. Since finding an equivalent description of the bisimulation relation is challenging, we can explore some necessary conditions. For instance, if two pointed relational models $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$ are sabotage bisimilar, then:

- $\mathcal{M}_1$ and $\mathcal{M}_2$ have the same number of edges.

– w_1 and w_2 have the same number of successors.
– w_1 is a self-loop if and only if w_2 is also a self-loop.
– w_1 returns to itself in at least n steps if and only if w_2 also returns to itself in at least n steps.

A reasonable combination of these necessary properties, forming a sub-problem, might help us obtain an ideal lower bound – this requires further investigation.

To end this chapter, let us get back to the general discussion of games on graphs. What does it mean to have a bisimulation between two game graphs with respect to two points on those two graphs? Evidently, whatever moves a player can make in one game, the same kind of moves can be made in the other game as well. Moreover, an alternation of the basic modality with the edge-deletion or node-deletion modality would describe a play in the game graphs with link deletion or point deletion, respectively. From the strategic viewpoint, the age-old copy strategy might be a relevant strategy to play on bisimilar game graphs. In fact, checking bisimilarity between different game graphs can be considered as a first step towards considering game-strategy equivalences between these games constituting structural changes in the underlying graphs. The related notion of $\langle up \rangle$-bisimulation contraction [30] concerns with the *simplest* graphs that are structurally equivalent to the original graphs. From the game perspectives, as mentioned earlier, we have the following question: What would the simplest graph in an equivalence class that would still show the essential structure of the game?

Acknowledgements. We thank Kaile Su for his comments and suggestions, which helped us to improve the chapter. Sujata Ghosh acknowledges Department of Science and Technology, Government of India for financial support vide Reference No DST/CSRI/2018/202 under Cognitive Science Research Initiative (CSRI) to carry out this work.

References

1. Luca Aceto, Anna Ingolfsdottir, and Jiri Srba. The algorithmics of bisimilarity. In *Advanced Topics in Bisimulation and Coinduction*, pages 100–172. Cambridge University Press, 2011.
2. Carlos Areces, Raul Fervari, and Guillaume Hoffmann. Moving arrows and four model checking results. In *International Workshop on Logic, Language, Information, and Computation*, pages 142–153. Springer, 2012.
3. Carlos Areces, Raul Fervari, and Guillaume Hoffmann. Swap logic. *Logic Journal of IGPL*, 22(2):309–332, 2014.
4. Carlos Areces, Diego Figueira, Santiago Figueira, and Sergio Mera. Expressive power and decidability for memory logics. In *Proceedings of the 15th International Workshop on Logic, Language, Information and Computation*, page 56–68. Springer-Verlag, 2008.
5. Guillaume Aucher, Johan van Benthem, and Davide Grossi. Modal logics of sabotage revisited. *Journal of Logic and Computation*, 28(2):269–303, 2018.
6. José Balcázar, Joaquim Gabarró, and Miklós Sántha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4(S1):638–648, nov 1992.
7. Alexandru Baltag, Dazhu Li, and Mina Young Pedersen. On the right path: A modal logic for supervised learning. In *International Workshop on Logic, Rationality and Interaction*, pages 1–14. Springer, 2019.
8. Alexandru Baltag and Lawrence S. Moss. Logics for epistemic programs. *Synthese*, 139(2):165–224, 2004.
9. Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge*, page 43–56. Morgan Kaufmann Publishers Inc., 1998.
10. Olav Bunte, Jan Friso Groote, Jeroen J. A. Keiren, Maurice Laveaux, Thomas Neele, Erik P. de Vink, Wieger Wesselink, Anton Wijs, and Tim A. C. Willemse. The mCRL2 toolset for analysing concurrent systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 21–39. Springer International Publishing, 2019.
11. Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, jan 1993.
12. Pierre Duchet and Henry Meyniel. Kernels in directed graphs: a poison game. *Discrete Mathematics*, 115(1-3):273–276, 1993.
13. Raul Fervari. *Relation-Changing Modal Logics*. PhD thesis, Universidad Nacional de Córdoba, 2014.
14. Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. CADP 2011: A toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer*, 15(2):89–107, 2013.
15. Sujata Ghosh, Shreyas Gupta, and Lei Li. Bisimulation in model-changing modal logics: An algorithmic study. *Journal of Logic and Computation*, 34:399–427, 2024.
16. Erich Grädel. Back and forth between logic and games. In *Lectures in Game Theory for Computer Scientists*, pages 99–145. Cambridge University Press, 2011.

17. Davide Grossi and Simon Rey. Credulous acceptability, poison games and modal logic. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, page 1994–1996. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2019.
18. John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. Introduction to automata theory, languages, and computation. *SIGACT News*, 32(1):60–65, mar 2001.
19. Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, page 228–240. Association for Computing Machinery, 1983.
20. Dazhu Li. Losing connection: The modal logic of definable link deletion. *Journal of Logic and Computation*, 30(3):715–743, 2020.
21. Christof Löding and Philipp Rohde. Model checking and satisfiability for sabotage modal logic. In *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science*, pages 302–313. Springer Berlin Heidelberg, 2003.
22. Christof Löding and Philipp Rohde. Solving the sabotage game is Pspace-hard. In *Mathematical Foundations of Computer Science 2003*, pages 531–540. Springer, Berlin, 2003.
23. Sergio Fernando Mera. *Modal Memory Logics*. PhD thesis, Université Henri Poincaré-Nancy 1, 2009.
24. Richard Nowakowski and Peter Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.
25. Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.
26. Philipp Rohde. *On Games and Logics over Dynamically Changing Structures*. PhD thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 2005.
27. Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.
28. Declan Thompson. Local fact change logic. In *Knowledge, Proof and Dynamics*, pages 73–96. Springer Singapore, 2020.
29. Johan van Benthem. An essay on sabotage and obstruction. In *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, pages 268–276. Springer, Heidelberg, 2005.
30. Johan van Benthem. *Modal Logic for Open Minds*. CSLI Publications, 2010.
31. Johan van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2011.
32. Johan van Benthem, Lei Li, Chenwei Shi, and Haoxuan Yin. Hybrid sabotage modal logic. *Journal of Logic and Computation*, 03 2022. exac006.
33. Johan van Benthem and Fenrong Liu. Graph games and logic design. In *Knowledge, Proof and Dynamics*, pages 125–146. Springer, Singapore, 2020.
34. Johan van Benthem, Krzysztof Mierzewski, and Francesca Zaffora Blando. The modal logic of stepwise removal. *The Review of Symbolic Logic*, page 1–28, 2020.
35. Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Springer, Dordrecht, 2008.
36. Francesca Zaffora Blando, Krzysztof Mierzewski, and Carlos Areces. The modal logics of the poison game. In *Knowledge, Proof and Dynamics*, pages 3–23. Springer Singapore, 2020.