Bisimulation in model-changing modal logics: An algorithmic study

Sujata Ghosh, Indian Statistical Institute, Chennai, India.
Shreyas Gupta, Indian Institute of Science, Bangalore, India.
Lei Li, Tsinghua University, Beijing, China.

Abstract

We discuss the notion of bisimulation in various model-changing modal logics and provide an algorithmic study of the same. We provide a general algorithm which gives an overall procedure to check whether two models are bisimilar in all these logics. Through our algorithmic analyses we provide a PSPACE upper bound of the bisimulation / model comparison problem of all these 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.

1 Introduction

Various model-changing modal logics have been introduced over the years to capture the model dynamics in many relevant areas, from mathematical systems to machine intelligence, from economic theories to philosophical queries as well as other important phenomena. For example, these logical systems deal with dynamical systems, knowledge and belief dynamics, graph dynamics, game dynamics, social network updates, memory updates and many others.

This whole study started with the introduction of Public Announcement Logic (*PAL*) that deals with information updates brought about by public communication. In *PAL*, as studied in [39, 40], the evaluation of the announcement formulas $\langle\varphi\rangle\psi$ (read as 'there is an announcement of φ after which ψ holds') involves deleting all the $\neg\varphi-$worlds in the Kripke models and all relations on those worlds, while in [19], the evaluation of the formula $[\varphi]_a\psi$ involves deleting all arrows in the Kripke models that are related to $\neg\varphi-$worlds, with the domain remaining the same. Dynamic Epistemic Logic (*DEL*)[10, 9, 54, 47], a generalization of *PAL*, characterizes such announcements in more subtle communications. Model-transforming operators like lexicographic upgrade $[\Uparrow]$, elite change $[\uparrow]$ and suggestion [#] capture plausibility relation updates under soft information [49, 46] involving relational changes without changes in the domain. Similar to *DEL*, Arrow Update Logic (*AUL*) [28] is a theory of epistemic access elimination that can be used to reason about multi-agent belief change, and furthermore, *GAUL*, a generalized version of *AUL*, characterizes similar dynamic changes as in *DEL*. In addition, factual changes are captured by updating valuations in Kripke models [41, 53, 52, 26].

Model-changing logics have also played a significant role in describing strategic reasoning in games on graphs, which have received a lot of attention in diverse domains, e.g., computer science, logic, linguistics, economics, mathematics, philosophy, biology and others. As the name indicates, such a game is played on directed or undirected graphs, and the players' actions are assigned based on the designer's research objectives. One can also consider different variants of such graph games where such variations can arise from different winning conditions (e.g., reachability, parity [20]), independent moves of players (e.g., cop and robber game [36]), one player obstructing moves of the others (e.g., sabotage game [45], poison game [14]) 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 perspective of link/edge deletion in graphs, sabotage games [45] 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 [50, 42, 30, 8, 48]. For example, the language of Definable Modal Sabotage Logic (S_dML) in [30] is a direct extension of basic modal language with additional operator $[-\psi]$. Intuitively, $[-\psi]\varphi$ expresses the condition that after local deletion/sabotage of all arrows from the current point, whose end points satisfy ψ, φ still holds. The Modal Logic of Supervised Learning (*SLL*) [8] is equipped with even more advanced sabotage operators that are introduced to characterize relation changes in multi-relation models. Moreover, there is a generalized sabotage operator in [48], denoted by $\blacklozenge^{\alpha}_{\beta}$ that is used to capture an arrow deletion whose end-points satisfy α and β, respectively.

A game that is close to the spirit of games describing point/vertex deletion on graphs is the poison game[14]: 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*[56, 22] use operators for changing valuations in and/or domains of models, which are inspired by memory logics[56, 34, 4]. Operators involving such changing of valuations are also mentioned in [43, 50]. Moreover, point-deletion style operators are proposed in [45, 15, 51, 2]. For example, the operator $\langle -\psi \rangle$ in the language of Modal Logic of Stepwise Removal (*MLSR*) [51] involves deleting $\psi-$worlds in the models.

In general, these logics aim to capture three mechanisms of model transformation, namely, those describing domain-changes, relation-changes and valuations-changes in models and their combinations. In this work, 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. We provide a uniform algorithmic study of the model comparison problem to shed some light on the complexity of these problems. For our purposes, we consider the operators $\langle sb \rangle$ and $\langle gsb \rangle$ [2, 5, 15, 48, 42] to model edge deletion in models, $\langle br \rangle$ and $\langle gbr \rangle$ [15] to model edge addition in models, and $\langle sw \rangle$ and $\langle gsw \rangle$ [3, 2, 15] to model arrow swap in models. In addition, we consider $\langle de \rangle$ [45] and $\langle ch \rangle$ [43] 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.

There is a strand of literature exploring technical properties of these logics. $MCML(\langle gsb \rangle)$ was first introduced in [45], and complete proof systems for $MCML(\langle gsb \rangle)$ have been discussed in [5, 15]. For the decidability and complexity questions, we have the following results from [33, 15, 43]: (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 $MCML(\langle up \rangle)$ 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 $MCML(\langle up \rangle)$ is PSPACE-complete. A result that is missing in this picture is the complexity of bisimulation or the model comparison problem, and in this work we investigate this issue by providing a uniform algorithmic study. The model comparison problem has been in the radar of researchers for a long time. In addition to the development of verification algorithms [13, 18, 21], model comparison problem also holds fundamental importance in the field of concurrency theory and related areas of computer science [24, 1]. It is well-known that deciding bisimilarity over finite labelled transition systems is in deterministic polynomial time [38, 25, 6]. 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 these model-changing logics mentioned. Finding lower bounds for these problems are left as open questions.

The rest of the paper can be summarized as follows: In section 2, we introduce the relevant logic frameworks together with their respective notions of bisimulations. Section 3 gives us a detailed algorithmic study together with providing upper bounds of the complexity of the relevant problems. Section 4 provides some further related results and concludes the paper with a discussion on the lower bound.

2 Model-changing modal logics

In this section, we first describe the various model-changing logics that we are going to base our study on. We will also recapitulate the corresponding notions of bisimulation. The main focus will be on the logics describing relation updates where the domains remain fixed. In addition, we will also look into domain updates as well as valuation updates. 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 $\mathcal{P}$, The syntax of the general model-changing modal logic $MCML(up)$ is given as follows:

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

where $p \in \mathcal{P}$, $\langle up\rangle$ 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 $V : W \to 2^{\mathcal{P}}$. 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, up-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 up-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 $p \in \mathcal{P}$.

(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 $(\mathcal{M}_1', u_1)$ such that $((\mathcal{M}_1, w_1), (\mathcal{M}_1', u_1)) \in r_{up}$, then there exists $(\mathcal{M}_2', u_2)$ such that $((\mathcal{M}_2, w_2), (\mathcal{M}_2', u_2)) \in r_{up}$ and $(\mathcal{M}_1', u_1)Z(\mathcal{M}_2', u_2)$.

(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 up-bisimulation between two pointed models $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$, we call them up-bisimilar.

Proposition 1. *If two pointed models* $(\mathcal{M}_1, w_1)$ *and* $(\mathcal{M}_2, w_2)$ *are* up*-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 up-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, 15], $\langle gsb\rangle$ appears in [2, 5, 15, 48, 42], $\langle sw\rangle$ appears in [3, 2, 15], $\langle gsw\rangle, \langle gbr\rangle$ are proposed in [15], $\langle de\rangle$ occurs in [45] and $\langle ch\rangle$ is proposed in [43] (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, $R_2 = \{(u, v) \in R_1 \mid u \neq w_1 \text{ and } v \neq w_1\}$, $V_2(u) = V_1(u)$ 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 $u \neq v$, where A is a set of proposition letters, and $w = v$.

Intuitively, the truth conditions of the above operators can be displayed in Fig1-8. For example, in Fig1, $\langle sb\rangle\varphi$ is true at $(\mathcal{M}_1, w)$, if and only if there exists pointed model $(\mathcal{M}_2, v)$ with $((\mathcal{M}_1, w), (\mathcal{M}_2, v)) \in r_{\langle sb\rangle}$ 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:

- If $(W_1, R_1, V_1, w_1)Z(W_2, R_2, V_2, w_2)$, then $(W_1, R_1, {V_1}_A^{w_1}, w_1)Z(W_2, R_2, {V_2}_A^{w_2}, w_2)$ for every $A \subseteq \mathcal{P}$, where for $i = 1, 2$, ${V_i}_A^{w_i}$ is almost V_i, except ${V_i}_A^{w_i} = A$.

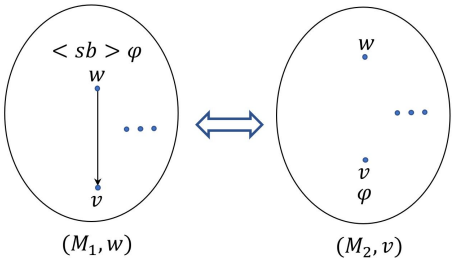


FIGURE 1: $\langle sb \rangle \varphi$

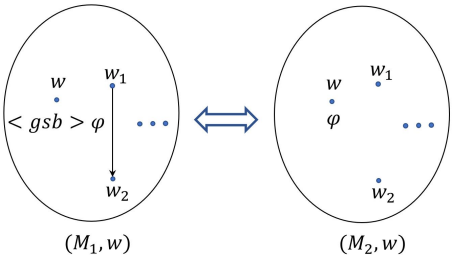


FIGURE 2: $\langle gsb \rangle \varphi$

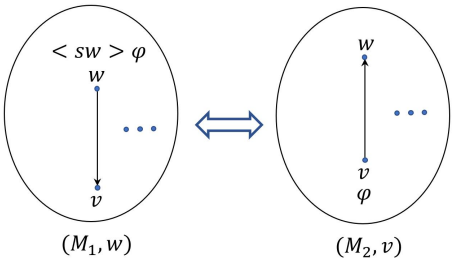


FIGURE 3: $\langle sw \rangle \varphi$

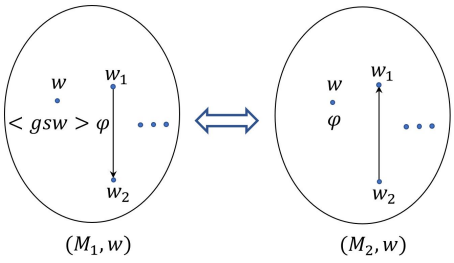


FIGURE 4: $\langle gsw \rangle \varphi$

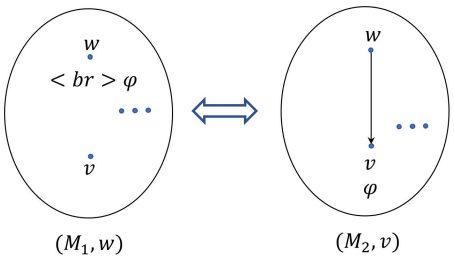


FIGURE 5: $\langle br \rangle \varphi$

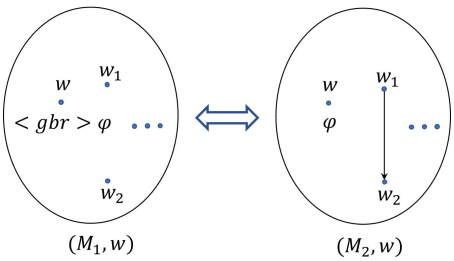


FIGURE 6: $\langle gbr \rangle \varphi$

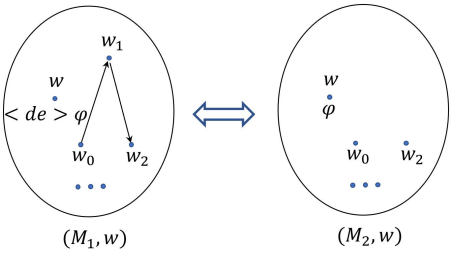


FIGURE 7: $\langle de \rangle \varphi$

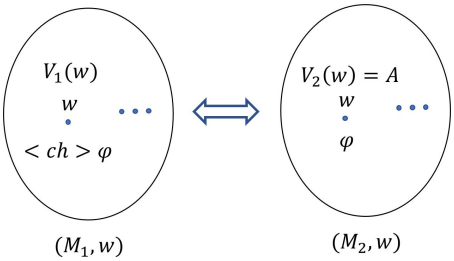


FIGURE 8: $\langle ch \rangle \varphi$

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

With these distinct notions of bisimulation, we are now all set to investigate the complexity of the following decision problem: Given two relational models, are they bisimilar?

3 An algorithmic study

Let us now provide algorithms to check whether two pointed models are up-bisimilar - we have eight distinct notions of bisimilarity based on different logics. Natural questions would be as follows: Do we need to have eight distinct algorithms or can we have a generalized one? How are these algorithms connected to each other? Can one be reduced to the other? We will first provide a general algorithm to check for bisimulation among models in all these logics and then move on to provide the same for the specific ones for a better understanding of the inherent connections/differences between various model-changing phenomena.

3.1 The Algorithm

In what follows, we provide a general algorithm (Algorithm 1). We define a function gen-Bisimilar 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 a state variable to specify the notion of bisimilarity that is to be checked. It outputs "Yes" if the two models are bisimlar, in the notion specified, and the function is called with $L = \emptyset$. All the notions of bisimilarity that we considered have 5 conditions to check. In the algorithm, we write a function to check these 5 conditions. Across different notions, conditions (1), (2) and (3) remain same. Therefore the only difference in the run of the algorithm for different notions comes in implementation of conditions (4) and (5). Now, one of the main problems that may come in implementation is when the given models have cycles. We have to check the successors for the (gen) bisimilarity too. This process may not terminate if 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 we have traveled. We initialize the algorithm with this list being empty, and keep adding edges that we have traveled before changing the models. We again make the list empty after the model changing step.

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.

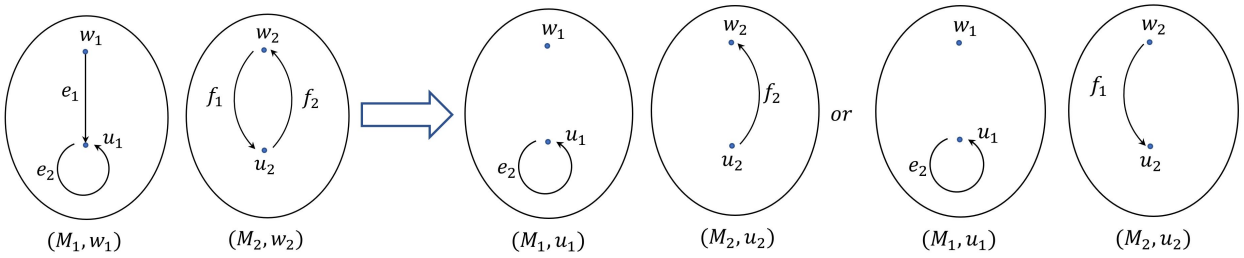


FIGURE 9: counterexample for $\langle gsb \rangle$-bisimilar

The Algorithm 1 takes input and passes the information, according to the value of $state$, to different algorithms for different checks. The first 3 conditions in the definition of $up-$bisimilarity remains same and hence algorithms 4 and 14 are always called. The 4th and 5th conditions change and accordingly different algorithms are called.

Algorithm 1 Algorithm to check whether two models are bisimilar in some model changing modal logic

Require: $((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2), L, state$

function GEN-BISIMILAR($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2), L, state$)
  if (state = $\langle gsb \rangle$-bisimilar OR $\langle gsw \rangle$-bisimilar) **then**
    if (checkEdges($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if
  if (state = $\langle de \rangle$-bisimilar OR $\langle gbr \rangle$-bisimilar) **then**
    if (checkNodes($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if
  if (checkAtomicPropositionInCurrentWorlds($(((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
    return No
  end if
  if (state = $\langle sb \rangle$-bisimilar) **then**
    if (checkEdgeDeletion($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if
  if (state = $\langle gsb \rangle$-bisimilar) **then**
    if (checkGeneralizedEdgeDeletion($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if
  if (state = $\langle sw \rangle$-bisimilar) **then**
    if (checkSwap($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if
  if (state = $\langle gsw \rangle$-bisimilar) **then**
    if (checkGeneralizedSwap($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if
  if (state = $\langle br \rangle$-bisimilar) **then**
    if (checkBridge($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if
  if (state = $\langle gbr \rangle$-bisimilar) **then**
    if (checkGeneralizedBridge($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if
  if (state = $\langle de \rangle$-bisimilar) **then**
    if (checkNodeDeletion($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
      return No
    end if
  end if

50: **if** (state = $\langle ch \rangle$-bisimilar) **then**
51: **if** (checkValuationChange($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2)$) = No) **then**
52: **return** No
53: **end if**
54: **end if**
55: **if** (checkSuccessors($((W_1, R_1, V_1), w_1), ((W_2, R_2, V_2), w_2), L, state$) = No) **then**
56: **return** No
57: **end if**
58: **return** Yes
59: **end function**

Now we will give a brief explanation of the different functions followed by the functions themselves. The function $checkEdges$ returns No if the input models do not have equal number of edges. This check makes the proof in section 3.2 a little easier.

Algorithm 2 checkEdges

1: **if** $|R_1| \neq |R_2|$ **then**
2: **return** No
3: **end if**
4: **return** Yes

The function $checkNodes$ returns No if the input models do not have equal number of nodes.

Algorithm 3 checkNodes

1: **if** $|W_1| \neq |W_2|$ **then**
2: **return** No
3: **end if**
4: **return** Yes

The function $checkAtomicPropositionInCurrentWorlds$ returns No if the nodes at which the input models are pointed, do not satisfy same set of atomic propositions. To do this, the algorithm checks that $w_1 \in V_1(p)$ if and only if $w_2 \in V_2(p)$, for all atomic propositions p.

Algorithm 4 checkAtomicPropositionInCurrentWorlds

1: **for** atomic propositions p **do**
2: **if** $(((w_1 \in V_1(p))$ & $(w_2 \notin V_2(p)))$ OR $((w_1 \notin V_1(p))$ & $(w_2 \in V_2(p))))$ **then**
3: **return** No
4: **end if**
5: **end for**
6: **return** Yes

The function $checkEdgeDeletion$ returns No, if there is no pair of models related to input models by relation $r_{\langle sb \rangle}$ that are $\langle sb \rangle$-bisimilar. To do this, the algorithm recursively calls the algorithm 1 on new models after deleting one edge from each (pointing from w_1). If all such instances of algorithm 1 return No, then by recursion, there is no pair of edges that can be deleted from given models that satisfies the conditions 4 and 5 in $\langle sb \rangle$-bisimilar definition.

Algorithm 5 checkEdgeDeletion

for $u_1 \in W_1$ **do**
Found = 0
  if $(w_1, u_1) \in R_1$ **then**
    for $u_2 \in W_2$ **do**
      if $(w_2, u_2) \in R_2$ **then**
        if gen-Bisimilar$(((W_1, R_1 \backslash \{(w_1, u_1)\}, V_1), u_1), ((W_2, R_2 \backslash \{(w_2, u_2)\}, V_2), u_2), \emptyset, state)$ = Yes **then**
          Found++
          break
        end if
      end if
    end for
    if Found = 0 **then**
      return No
    end if
  end if
end for
for $u_2 \in W_2$ **do**
Found = 0;
  if $(w_2, u_2) \in R_2$ **then**
    for $u_1 \in W_1$ **do**
      if $(w_2, u_2) \in R_2$ **then**
        if gen-Bisimilar$(((W_1, R_1 \backslash \{(w_1, u_1)\}, V_1), u_1), ((W_2, R_2 \backslash \{(w_2, u_2)\}, V_2), u_2), \emptyset, state)$ = Yes **then**
          Found++
          break
        end if
      end if
    end for
    if Found = 0 **then**
      return No
    end if
  end if
end for
return Yes

The function *checkGeneralizedEdgeDeletion* returns No, if there is no pair of models related to the input models by relation $r_{\langle gsb \rangle}$ that are $\langle gsb \rangle$-bisimilar. This algorithm works very similar to the previous one with the only difference being as follows - instead of deleting edges pointed from w_1, it runs over all edges. This is in accordance with the difference in definitions of $\langle sb \rangle$-bisimilarity and $\langle gsb \rangle$-bisimilarity.

Algorithm 6 checkGeneralizedEdgeDeletion

for $e_1 \in R_1$ **do**
Found = 0
  for $e_2 \in R_2$ **do**
    if gen-Bisimilar$(((W_1, R_1 \backslash \{e_1\}, V_1), w_1), ((W_2, R_2 \backslash \{e_2\}, V_2), w_2), \emptyset, state)$ = Yes **then**
      Found++
      break
    end if
  end for

```
    if Found = 0 then
        return No
    end if
end for
for e_2 ∈ R_2 do
Found = 0
    for e_1 ∈ R_1 do
        if gen-Bisimilar(((W_1, R_1\{e_1}, V_1), w_1), ((W_2, R_2\{e_2}, V_2), w_2), ∅, state) = Yes then
            Found++
            break
        end if
    end for
    if Found = 0 then
        return No
    end if
end for
return Yes
```

The function $checkSwap$ returns No, if there is no pair of models related to the input models by relation $r_{\langle sw \rangle}$ that are $\langle sw \rangle$-bisimilar. To do this, it runs over all the edges from w_1 and w_2, swaps their directions, and calls algorithm 1 recursively.

Algorithm 7 checkSwap

```
for u_1 ∈ W_1 do
Found = 0
    if (w_1, u_1)∈ R_1\L' then
        for u_2 ∈ W_2 do
            if (w_2, u_2)∈ R_2\L' then
                if gen-Bisimilar(((W_1, R_1∪{(u_1, w_1)}\{(w_1, u_1)}, V_1), u_1),((W_2, R_2∪{(u_2, w_2)}\{(w_2, u_2)}, V_2), u_2),
    L' ∪ {(w_1, u_1), (w_2, u_2)}, state) = Yes then
                    Found++
                    break
                end if
            end if
        end for
        if Found = 0 then
            return No
        end if
    end if
end for
for u_2 ∈ W_2 do
Found = 0
    if (w_2, u_2)∈ R_2\L' then
        for u_1 ∈ W_1 do
            if (w_1, u_1)∈ R_1\L' then
                if gen-Bisimilar(((W_1, R_1∪{(u_1, w_1)}\{(w_1, u_1)}, V_1), u_1),((W_2, R_2∪{(u_2, w_2)}\{(w_2, u_2)}, V_2), u_2),
    L' ∪ {(w_1, u_1), (w_2, u_2)}, state) = Yes then
                    Found++
                    break
                end if
```

```
            end if
        end for
        if Found = 0 then
            return No
        end if
    end if
end for
return Yes
```

The function $checkGeneralizedSwap$ returns No, if there is no pair of models related to the input models by relation $r_{\langle gsw \rangle}$ that are $\langle gsw \rangle$-bisimilar. Again, this is very similar to the previous algorithm with the only difference being that it now runs over all edges.

Algorithm 8 checkGeneralizedSwap

```
for (u1, u2) ∈ R1\L' do
Found = 0;
    for (v1, v2) ∈ R2\L' do
        if gen-Bisimilar(((W1, R1 ∪ {(u2, u1)}\{(u1, u2)}, V1), w1),((W2, R2 ∪ {(v2, v1)}\{(v1, v2)}, V2), w2), L' ∪
{(u1, u2), (v1, v2)}, state) = Yes then
            Found++
            break
        end if
    end for
    if Found = 0 then
        return No
    end if
end for
for (v1, v2) ∈ R2\L' do
Found = 0;
    for (u1, u2) ∈ R1\L' do
        if gen-Bisimilar(((W1, R1 ∪ {(u2, u1)}\{(u1, u2)}, V1), w1),((W2, R2 ∪ {(v2, v1)}\{(v1, v2)}, V2), w2), L' ∪
{(u1, u2), (v1, v2)}, state) = Yes then
            Found++
            break
        end if
    end for
    if Found = 0 then
        return No
    end if
end for
return Yes
```

The function $checkBridge$ returns No, if there is no pair of models related to the input models by relation $r_{\langle br \rangle}$ that are $\langle br \rangle$-bisimilar. To do this, the algorithm adds new edges from w_1 and w_2 and calls algorithm 1 recursively.

Algorithm 9 checkBridge

```
for u1 ∈ W1 do
Found = 0
    for u2 ∈ W2 do
```

```
        if (((w1, u1) ∉ R1) & ((w2, u2) ∉ R2) then
            if gen-Bisimilar(((W1, R1 ∪ {(w1, u1)}, V1), u1),((W2, R2 ∪ {(w2, u2)}, V2), u2), ∅, state) = Yes then
                Found++
                break
            end if
        end if
    end for
    if Found = 0 then
        return No
    end if
end for
for u2 ∈ W2 do
Found = 0
    for u1 ∈ W1 do
        if (((w1, u1) ∉ R1) & ((w2, u2) ∉ R2) then
            if gen-Bisimilar(((W1, R1 ∪ {(w1, u1)}, V1), u1),((W2, R2 ∪ {(w2, u2)}, V2), u2), ∅, state) = Yes then
                Found++
                break
            end if
        end if
    end for
    if Found = 0 then
        return No
    end if
end for
return Yes
```

The function $checkGeneralizedBridge$ returns No, if there is no pair of models related to the input models by relation $r_{\langle gbr\rangle}$ that are $\langle gbr\rangle$-bisimilar. This algorithm adds one new edge to both the models and calls algorithm 1 recursively.

Algorithm 10 checkGeneralizedBridge

```
for (u1, u2) ∈ W1 × W1 do
Found = 0
    for (v1, v2) ∈ W2 × W2 do
        if (((u1, u2) ∉ R1) & ((v1, v2) ∉ R2) then
            if gen-Bisimilar(((W1, R1 ∪ {(u1, u2)}, V1), w1),((W2, R2 ∪ {(v1, v2)}, V2), w2), ∅, state) = Yes then
                Found++
                break
            end if
        end if
    end for
    if Found = 0 then
        return No
    end if
end for
for (v1, v2) ∈ W2 × W2 do
Found = 0
    for (u1, u2) ∈ W1 × W1 do
        if (((u1, u2) ∉ R1) & ((v1, v2) ∉ R2) then
```

```
            if gen-Bisimilar(((W1, R1 ∪ {(u1, u2)}, V1), w1),((W2, R2 ∪ {(v1, v2)}, V2), w2), ∅, state) = Yes then
                Found++
                break
            end if
        end if
    end for
    if Found = 0 then
        return No
    end if
end for
return Yes
```

The previous algorithms were the implementations of the model-changing step for the six relation-changing logics that we have described. We now move on to implement the model-changing step for domain-changing logic and valuation-changing logic. The next algorithm is a precursor to the case of domain-changing logic. Specifically, it computes the new model after a node has been deleted from it.

Algorithm 11 algorithm to compute new relational model after point deletion

Require: $((W, R, V), w), u$

```
function SUCCESSOR(((W, R, V), w), u)
W' = W\{u}
R' = R\({(u, v) ∈ W | v ∈ R} ∪ {(v, u) ∈ R | v ∈ W})
    for p ∈ P do
        V'(p) = V(p) ∩ W'
    end for
    return ((W', R', V'), w)
end function
```

The function $checkNodeDeletion$ returns No, if there is no pair of models related to the input models by relation $r_{\langle de \rangle}$ that are $\langle de \rangle$-bisimilar. To do this, the algorithm makes a recursive call to $gen - Bisimilar$ with new pair of models that have one node deleted in each.

Algorithm 12 checkNodeDeletion

```
for u1 ∈ W1 do
Found = 0
    for u2 ∈ W2 do
        if (u1 ≠ w1) & (u2 ≠ w2) then
            if gen-Bisimilar((successor((W1, R1, V1), u1), w1),(successor((W2, R2, V2), u2), w2), ∅, state) = Yes
then
                Found++
                break
            end if
        end if
    end for
    if Found = 0  & (u1 ≠ w1) then
        return No
    end if
end for
for u2 ∈ W2 do
```

Found = 0
    for $u_1 \in W_1$ **do**
        if $(u_1 \neq w_1)$ & $(u_2 \neq w_2)$ **then**
            if gen-Bisimilar($(successor((W_1, R_1, V_1), u_1), w_1)$,$(successor((W_2, R_2, V_2), u_2), w_2), \emptyset, state$) = Yes **then**
                Found++
                break
            end if
        end if
    end for
    if Found = 0 & $(u_2 \neq w_2)$ **then**
        return No
    end if
end for
return Yes

The function $checkValuationChange$ returns No, if there is no pair of models related to the input models by relation $r_{\langle ch \rangle}$ that are $\langle ch \rangle$-bisimilar. To do this, the function changes valuation of the current node and then calls $gen - Bisimilar$ recursively.

Algorithm 13 checkValuationChange

for $A \subset \mathcal{P}$ **do**
    $\tilde{V_1}(w_1) = V_1(w)$
end for
for $w \in W_1 \backslash \{w_1\}$ **do**
    $\tilde{V_1}(w) = V_1(w)$
end for
for $A \subset \mathcal{P}$ **do**
    $\tilde{V_2}(w_2) = A$
end for
for $w \in W_2 \backslash \{w_2\}$ **do**
    $\tilde{V_2}(w) = V_2(w)$
end for
if gen-Bisimilar($((W_1, R_1, \tilde{V_1}), w_1)$,$((W_2, R_2, \tilde{V_2}), w_2), L, state$) = No **then**
    return No
end if
return Yes

The following algorithm checks for the existence of successors to the node, at which the input models are pointed, for the specific bisimulation according to the state. Specifically, it checks if conditions 2 and 3 in the definition of $up-$bisimilarity are true. To do this, it changes the node where the models are pointed to one of the successors of initial nodes at which the models were pointed, and then makes a recursive call to $gen - Bisimilar$.

Algorithm 14 checkSuccessors

if $(w_1, w_2) \notin L$ **then**
    for $u_1 \in W_1$ **do**
        Found = 0
        for $u_2 \in W_2$ **do**
            if $((w_1 R_1 u_1)$ & $(w_2 R_2 u_2))$ **then**

```
                if ((u1, u2) ∉ L) then
                    if gen-Bisimilar(((W1, R1, V1), u1), ((W2, R2, V2), u2), L ∪ {(w1, w2)}, state) = Yes then
                        Found++
                    end if
                else
                    Found++
                end if
            end if
        end for
        if (Found = 0) & (w1 R1 u1) then
            return No
        end if
    end for
    for u2 ∈ W2 do
        Found = 0
        for u1 ∈ W1 do
            if ((w1 R1 u1) & (w2 R2 u2)) then
                if ((u1, u2) ∉ L) then
                    if gen-Bisimilar(((W1, R1, V1), u1), ((W2, R2, V2), u2), L ∪ {(w1, w2)}, state) = Yes then
                        Found++
                    end if
                else
                    Found++
                end if
            end if
        end for
        if (Found = 0) & (w2 R2 u2) then
            return No
        end if
    end for
end if
return Yes
```

3.2 On $\langle gsb \rangle$-bisimulation

We will now give a detailed proof that the algorithm works when $state = \langle gsb \rangle - bisimilar$. The other cases can be proved in a similar manner. Before going into the main proof, we have the following lemma.

Lemma 2. *If* $((W_1, R_1, V_1), w_1) \underline{\leftrightarrow}_s ((W_2, R_2, V_2), w_2)$ *and* $|R_1|$ *and* $|R_2|$ *are finite, then* $|R_1| = |R_2|$.

Proof. Suppose on the contrary, $|R_1| \neq |R_2|$. Without loss of generality, assume $|R_1| < |R_2|$.
Proof by induction on $n = |R_1|$

- **Base case:** $n = 0$
  By assumption $|R_1| = 0$ and $|R_2| > 0$. So $\exists e \in R_2$. Now, since $((W_1, R_1, V_1), w_1) \underline{\leftrightarrow}_s ((W_2, R_2, V_2), w_2)$, they satisfy condition (5) of the definition of $\langle gsb \rangle-$bisimilarity. Therefore, there must exist an edge $f \in R_1$ such that $((W_1, R_1 \backslash \{f\}, V_1), w_1) \underline{\leftrightarrow}_s ((W_2, R_2 \backslash \{e\}, V_2), w_2)$. But, since $|R_1| = 0$, no such f can exist. Contradiction.
- **Induction hypothesis:** Suppose the claim holds good for $n \leq k$, *i.e.*, $|R_1| = |R_2|$, whenever $|R_1| \leq k$.
- **Induction step:** $n = k + 1$
  Suppose $\min(|R_1|, |R_2|) = |R_1| = k+1$. Let $e_1 \in R_1$ be any edge. Since, $((W_1, R_1, V_1), w_1) \underline{\leftrightarrow}_s ((W_2, R_2, V_2), w_2)$, they satisfy condition (4) in the definition of $\langle gsb \rangle-$bisimilarity, so $\exists e_2 \in R_2$ such that $((W_1, R_1 \backslash \{e_1\}, V_1), w_1) \underline{\leftrightarrow}_s ((W_2, R_2 \backslash \{e_2\}, V_2), w_2)$. But then by induction hypothesis, we have $|R_1 \backslash \{e_1\}| = R_2 \backslash \{e_2\}| \implies |R_1| - 1 = |R_2| - 1 \implies |R_1| = |R_2|$.

This completes the proof. □

Theorem 3. *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* gen-Bisimilar$((\mathcal{M}_1, w_1), (\mathcal{M}_2, w_2), \emptyset, \langle gsb \rangle - bisimilar)$ *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. Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ have different number of edges, then the function returns No at line 2 in algorithm 2, and $(\mathcal{M}_1, w_1) \underline{\not\leftrightarrow}_s (\mathcal{M}_2, w_2)$. So, let us consider that both models have equal number of edges (say n). We prove by induction on n:

> **Base case:** $n = 0$.
To prove $(\mathcal{M}_1, w_1) \underline{\leftrightarrow}_s (\mathcal{M}_2, w_2)$ iff the function returns yes when $R_1 = \emptyset = R_2$. We will first prove, by contrapositivity, that if the function returns yes, then $(\mathcal{M}_1, w_1) \underline{\leftrightarrow}_s (\mathcal{M}_2, w_2)$.

> > Suppose $(\mathcal{M}_1, w_1) \underline{\not\leftrightarrow}_s (\mathcal{M}_2, w_2)$. Then they violate one of the five conditions in the definition of $\langle gsb \rangle-$bisimilarity (in Section 2.1).

> > > Suppose they violate condition (1). Then there is some atomic proposition p such that either $(\mathcal{M}_1, w_1) \models p$ and $(\mathcal{M}_2, w_2) \not\models p$; or $(\mathcal{M}_1, w_1) \not\models p$ and $(\mathcal{M}_2, w_2) \models p$. From truth definitions, we have $w_1 \in V_1(p)$ but $w_2 \notin V_2(p)$; or $w_1 \notin V_1(p)$ but $w_2 \in V_2(p)$. In this case, the function returns No at line 3 in algorithm 4.

> > > Suppose they violate condition (2). Then, there is a successor v_1 of w_1, i.e. $\exists v_1 \in W_1$ such that $w_1 R_1 v_1$, but $\forall v_2$ such that $w_2 R_2 v_2$, we do not have $(\mathcal{M}_1, v_1) \underline{\leftrightarrow}_s (\mathcal{M}_2, v_2)$. But since $n = 0$, $w_1 R_1 v_1$ does not hold for any v_1 as $R_1 = \emptyset$. Therefore, condition (2) in the definition of $\langle gsb \rangle-$bisimilarity cannot be violated in this case.

> > > Suppose that they violate condition (3). Again by similar argument as last point, we can not have $v_2 R_2 w_2$ and hence condition (3) can not be violated when $n = 0$.

> > > Suppose they violate condition (4). Then there is an edge $e_1 \in R_1$ such that for any edge $e_2 \in R_2$, it is not the case that $(\mathcal{M}_1 \backslash \{e_1\}, w_1) \underline{\leftrightarrow}_s (\mathcal{M}_2 \backslash \{e_2\}, w_2)$. But again since $n = 0$, $R_1 = \emptyset$, hence no such e_1 exists. So this case cannot arise.

> > > By similar argument as in previous point, the models cannot violate condition (5).

> > Now we will prove the other side. Therefore, suppose that the function returns No, Then one of the following cases occur:

> > > The function returns No at line number 3 in algorithm 4. This can only happen when the If condition at line 2 in algorithm 4 is true. Therefore, there exists an atomic proposition p such that, $w_1 \in V_1(p)$ but $w_2 \notin V_2(p)$; or $w_1 \notin V_1(p)$ but $w_2 \in V_2(p)$. From truth definitions, we have either $(\mathcal{M}_1, w_1) \models p$ and $(\mathcal{M}_2, w_2) \not\models p$; or $(\mathcal{M}_1, w_1) \not\models p$ and $(\mathcal{M}_2, w_2) \models p$. But then $(\mathcal{M}_1, w_1) \underline{\not\leftrightarrow}_s (\mathcal{M}_2, w_2)$ as they violate condition (1) of the definition of $\langle gsb \rangle-$bisimilarity.

> > > The function returns No at line number 10 or 22 in algorithm 6. But since $R_1 = \emptyset$ and $R_2 = \emptyset$, This can not happen as the function $checkGeneralizedEdgeDeletion$ will not execute any for loop.

> > > Suppose the function returns No from line 16 or 33 in algorithm 14. Again, this can not happen because w_1 and w_2 do not have any successors.

This completes both sides of the base case.

> **Induction Hypothesis 1:** Suppose the theorem holds for $n \leq k$. That is, $(\mathcal{M}_1, w_1) \underline{\leftrightarrow}_s (\mathcal{M}_2, w_2)$ iff the function gen-Bisimilar$((\mathcal{M}_1, w_1), (\mathcal{M}_2, w_2), \emptyset, \langle gsb \rangle - bisimilar)$ returns yes when $|R_1| = |R_2| \leq k$.

> **Induction Step:** Let $n = k + 1$
We will first prove that if $(\mathcal{M}_1, w_1) \underline{\leftrightarrow}_s (\mathcal{M}_2, w_2)$ then the function returns yes. Again we will prove this by contrapositivity.

> > Suppose the function returns No in algorithm 1. Then it executes one of the 4 return No statements, that are reachable when $state = \langle gsb \rangle - bisimilar$. But it can not return No at line 2 in algorithm 2, as we have assumed $|R_1| = |R_2|$. So the following cases can occur:

> > > The function returns No at line number 13 in algorithm 1. This can only happen when the If condition at line 2 in algorithm 4 is true. But then, by argument similar to that in base case, $(\mathcal{M}_1, w_1) \not\leftrightarrows_s (\mathcal{M}_2, w_2)$ as they violate condition (1) of the definition of $\langle gsb \rangle$-bisimilarity.

> > > The function returns No at line number 22 in algorithm 1. Then condition in line 21 is true. This happens if the function $checkGeneralizedEdgeDeletion$ returns No at line 10 or 22. If the function returns No at line 10 in function $checkGeneralizedEdgeDeletion$, there is an $e_1 \in R_1$ such that for all $e_2 \in R_2$, we have gen-bisimilar$((\mathcal{M}_1 \backslash \{e_1\}, w_1), (\mathcal{M}_2 \backslash \{e_2\}, w_2), \emptyset, \langle gsb \rangle - bisimilar)$ returns No. But the model $\mathcal{M}_1' = \mathcal{M}_1 \backslash \{e_1\}$ and $\mathcal{M}_2' = \mathcal{M}_2 \backslash \{e_2\}$ have k edges. Therefore, by induction hypothesis 1, $(\mathcal{M}_1 \backslash \{e_1\}, w_1) \not\leftrightarrows_s (\mathcal{M}_2 \backslash \{e_2\}, w_2)$ for all $e_2 \in R_2$. This is violation to condition (4) in the definition of $\langle gsb \rangle$-bisimilarity. Therefore, $(\mathcal{M}_1, w_1) \not\leftrightarrows_s (\mathcal{M}_2, w_2)$. The case is similar if No is returned at line 22 in function $checkGeneralizedEdgeDeletion$.

> > > The function returns No at line 56 in algorithm 1. Then condition at line 15 or 32 in algorithm 14 in the function $checkSuccessors$ is true for some $u_1 \in W_1$ or $u_2 \in W_2$ respectively. Suppose the function returns No at line 16. Therefore, following cases arise (following line numbers are in function $checkSuccessors$):

> > > > For a successor u_1 of w_1, condition at line 5 is false for all $u_2 \in W_2$, $i.e.$, $w_2 R_2 u_2$ is not true for any $u_2 \in W_2$. This is a violation of condition (2) in definition of $\langle gsb \rangle$−bisimilarity and hence $(\mathcal{M}_1, w_1) \not\leftrightarrows_s (\mathcal{M}_2, w_2)$

> > > > Condition at line 5 and 6 are true but condition at line 7 is false, $i.e.$, $\exists u_1 \in W_1$ such that $w_1 R_1 u_1$, $\forall u_2 \in R_2$ such that $w_2 R_2 u_2$ and L is such that $(u_1, u_2) \notin L$ (and $(w_1, w_2) \notin L$ because line 7 can be executed only if condition in line 1 is true); we get gen-Bisimilar$((\mathcal{M}_1, u_1), (\mathcal{M}_2, u_2), L \cup \{(w_1, w_2)\}, \langle gsb \rangle - bisimilar)$ returns No.
To prove:$(\mathcal{M}_1, w_1) \not\leftrightarrows_s (\mathcal{M}_2, w_2)$.
Proof by induction on $m = |W_1 \times W_2| - |L \cup \{(w_1, w_2)\}|$

> > > > > **Base case:** $|W_1 \times W_2| = |L \cup \{(w_1, w_2)\}|$
We need to prove that if $\exists u_1 \in W_1$ such that $w_1 R_1 u_1$, $\forall u_2 \in R_2$ such that $w_2 R_2 u_2$ and L is such that $(u_1, u_2) \notin L$ (and $(w_1, w_2) \notin L$ because line 5 can be executed only if condition in line 1 is true) and $|W_1 \times W_2| - |L \cup \{(w_1, w_2)\}| = 0$; and gen-Bisimilar$((\mathcal{M}_1, u_1), (\mathcal{M}_2, u_2), L \cup \{(w_1, w_2)\}, \langle gsb \rangle - bisimilar)$ returns No, then $(\mathcal{M}_1, w_1) \not\leftrightarrows_s (\mathcal{M}_2, w_2)$.
But since $|W_1 \times W_2| - |L \cup \{(w_1, w_2)\}| = 0$, we have $(u_1, u_2) \in L \cup \{(w_1, w_2)\}$. This is in contradiction with condition in line 6 being true. So the antecedent is false and hence base case is true vacuously.

> > > > > **Induction Hypothesis 2:** Suppose the claim holds for $m \leq l$, $i.e.$,
Suppose whenever $\exists u_1 \in W_1$ such that $w_1 R_1 u_1$, $\forall u_2 \in R_2$ such that $w_2 R_2 u_2$ and L is such that $(u_1, u_2) \notin L$ (and $(w_1, w_2) \notin L$ because line 7 can be executed only if condition in line 1 is true) and $|W_1 \times W_2| - |L \cup \{(w_1, w_2)\}| \leq l$; and gen-Bisimilar$((\mathcal{M}_1, u_1), (\mathcal{M}_2, u_2), L \cup \{(w_1, w_2)\}, \langle gsb \rangle - bisimilar)$ returns No, then $(\mathcal{M}_1, w_1) \not\leftrightarrows_s (\mathcal{M}_2, w_2)$

> > > > > **Induction step:** Suppose $m = l + 1$.
In this case, suppose condition in line 6 true and condition in line 7 is false. Therefore, we have, $\exists u_1 \in W_1$ such that $w_1 R_1 u_1$, $\forall u_2 \in R_2$ such that $w_2 R_2 u_2$ and L is such that $(u_1, u_2) \notin L$ (and $(w_1, w_2) \notin L$ because line 6 can be executed only if condition in line 1 is true) and $|W_1 \times W_2| - |L \cup \{(w_1, w_2)\}| = l + 1$; and gen-Bisimilar$((\mathcal{M}_1, u_1), (\mathcal{M}_2, u_2), L \cup \{(w_1, w_2)\}, \langle gsb \rangle - bisimilar)$ returns No. Now, gen-Bisimilar$((\mathcal{M}_1, u_1), (\mathcal{M}_2, u_2), L \cup \{(w_1, w_2)\}, \langle gsb \rangle - bisimilar)$ can return No either at one of 6 , reachable return No statements in gen-bisimilar. If it returns No at first 4, then by above cases, we have already proved that $(\mathcal{M}_1, u_1) \not\leftrightarrows_s (\mathcal{M}_2, u_2)$ because they violate conditions (1) or (4) or (5) in the definition of $\langle gsb \rangle$−bisimilarity. Suppose it returns No at line 56 in gen-bisimilar, then the function $checkSuccessors$ returns No at line 16 or 33. Assume it returns No at line 16. Then if condition at line 5 is always false, then $(\mathcal{M}_1, u_1) \not\leftrightarrows_s (\mathcal{M}_2, u_2)$ because they violate condition (2) of definition of $\langle gsb \rangle$−bisimilarity. So suppose condition at line 6 is true but at line 7 is false. Therefore, $\exists v_1 \in W_1$ such that $u_1 R_1 v_1$ and $\forall v_2 \in W_2$ such that $u_2 R_2 v_2$, L is such that $(v_1, v_2) \notin L \cup \{(w_1, w_2)\}$, we have gen-Bisimilar$((\mathcal{M}_1, v_1), (\mathcal{M}_2, v_2), L \cup \{(w_1, w_2), (u_1, u_2)\}, \langle gsb \rangle - bisimilar)$ returns No. Now by induction hypothesis 2, $(\mathcal{M}_1, v_1) \not\leftrightarrows_s (\mathcal{M}_2, v_2)$ which implies $(\mathcal{M}_1, u_1) \not\leftrightarrows_s (\mathcal{M}_2, u_2)$ and hence $(\mathcal{M}_1, w_1) \not\leftrightarrows_s (\mathcal{M}_2, w_2)$.

> > > The function returns No at line 33 in algorithm 14, then by argument similar to last case, $(\mathcal{M}_1, w_1) \not\leftrightarrows_s (\mathcal{M}_2, w_2)$.
We will now prove the remaining side by contrapositivity.

> Suppose $(\mathcal{M}_1, w_1) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, w_2)$. Then these models must violate one of the 5 conditions in definition of $\langle gsb \rangle-$bisimilarity.

> > Suppose they violate condition (1). There there is some atomic proposition p such that either $(\mathcal{M}_1, w_1) \models p$ and $(\mathcal{M}_2, w_2) \not\models p$; or $(\mathcal{M}_1, w_1) \not\models p$ and $(\mathcal{M}_2, w_2) \models p$. From truth definitions, we have $w_1 \in V_1(p)$ but $w_2 \notin V_2(p)$; or $w_1 \notin V_1(p)$ but $w_2 \in V_2(p)$. In this case the function returns No in line 13.

> > Suppose they violate condition (4). Then there is an edge $e_1 \in R_1$ such that for any edge $e_2 \in R_2$, $(\mathcal{M}_1 \backslash \{e_1\}, w_1) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2 \backslash \{e_2\}, w_2)$. In this case for e_1, condition in line 4 in algorithm 6 is never true. By induction hypothesis 1 ($\mathcal{M}_1 \backslash \{e_1\}$ and $\mathcal{M}_2 \backslash \{e_2\}$ have k edges, hence we can use induction hypothesis). Therefore, return No is executed in line 10 in function $checkGeneralizedEdgeDeletion$.

> > Suppose they violate condition (5), by similar argument as previous case, by induction hypothesis, function returns No.

> > Suppose they violate condition (2) and/ or (3). We must prove if $(\mathcal{M}_1, w_1) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, w_2)$ because they violate condition (2) and/or (3), but not (1), (4) or (5) in the definition of $\langle gsb \rangle-$bisimilarity, then gen-Bisimilar($((\mathcal{M}_1, w_1), (\mathcal{M}_2, w_2), \emptyset, \langle gsb \rangle - bisimilar$) returns No.

Since $(\mathcal{M}_1, w_1) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, w_2)$ because they violate condition (2) and/or (3), therefore $\exists u_{11} \in W_1$, $w_1 R_1 u_{11}$, such that $\forall u_{21} \in W_2$, $w_2 R_2 u_{21}$, $(\mathcal{M}_1, u_{11}) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, u_{21})$ (if condition (2) is violated); or $\exists u_{12} \in W_2$, $w_2 R_2 u_{12}$, such that $\forall u_{11} \in W_1$, $w_1 R_1 u_{11}$, $(\mathcal{M}_1, u_{11}) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, u_{21})$. Now if $(\mathcal{M}_1, u_{11}) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, u_{21})$ because they violate conditions (1), (4) or (5), then by previous cases, the function returns No and we will be done. Let us pick a general such pair (v_{11}, v_{21}). So, assume $(\mathcal{M}_1, v_{11}) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, v_{21})$ because they violate condition(s) (2) and/or (3). Therefore, again, $\exists u_{12} \in W_1$, $v_{11} R_1 u_{12}$, such that $\forall u_{22} \in W_2$, $v_{12} R_2 u_{22}$, $(\mathcal{M}_1, u_{12}) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, u_{22})$ (if they violate (2)); or $\exists u_{22} \in W_2$, $v_{12} R_2 u_{22}$, such that $\forall u_{12} \in W_1$, $v_{11} R_1 u_{12}$, $(\mathcal{M}_1, u_{12}) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, u_{22})$ (if they violate condition (3). Again, choose a general such pair (v_{12}, v_{22}) from above such that $v_{11} R_1 v_{12}$ and $v_{21} R_2 v_{22}$ and $(\mathcal{M}_1, v_{12}) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, v_{22})$. Again, we are done if $(\mathcal{M}_1, v_{12}) \underleftrightarrow{\not\leftrightarrow}_s (\mathcal{M}_2, v_{22})$ because they violate condition (1), (4) or (5). So, again, we can assume that they violate condition (2) and /or (3). This can go on until we reach a leaf node, $i.e.$, there is some k such that exactly one of the following is true: $v_{1k} R_1 v_{1(k+1)}$ for some $v_{1(k+1)} \in W_1$ or $v_{2k} R_2 v_{2(k+1)}$ for some $v_{2(k+1)} \in W_2$. Again the function returns No, from function $checkGeneralizedEdgeDeletion$ in both cases. The only case that remains is when there is no leaf nodes and there is some k such that $v_{1k} = v_{1l}$ or w_1 and $v_{2k} = v_{2l}$ or w_2 for some $l < k$. In this case, since v_{1i} and v_{2i} were some general node in the reachable part from w_1 and w_2, such that they do not violate condition (1), (4) or (5) in the definition of $\langle gsb \rangle-$bisimilarity, we have the following:

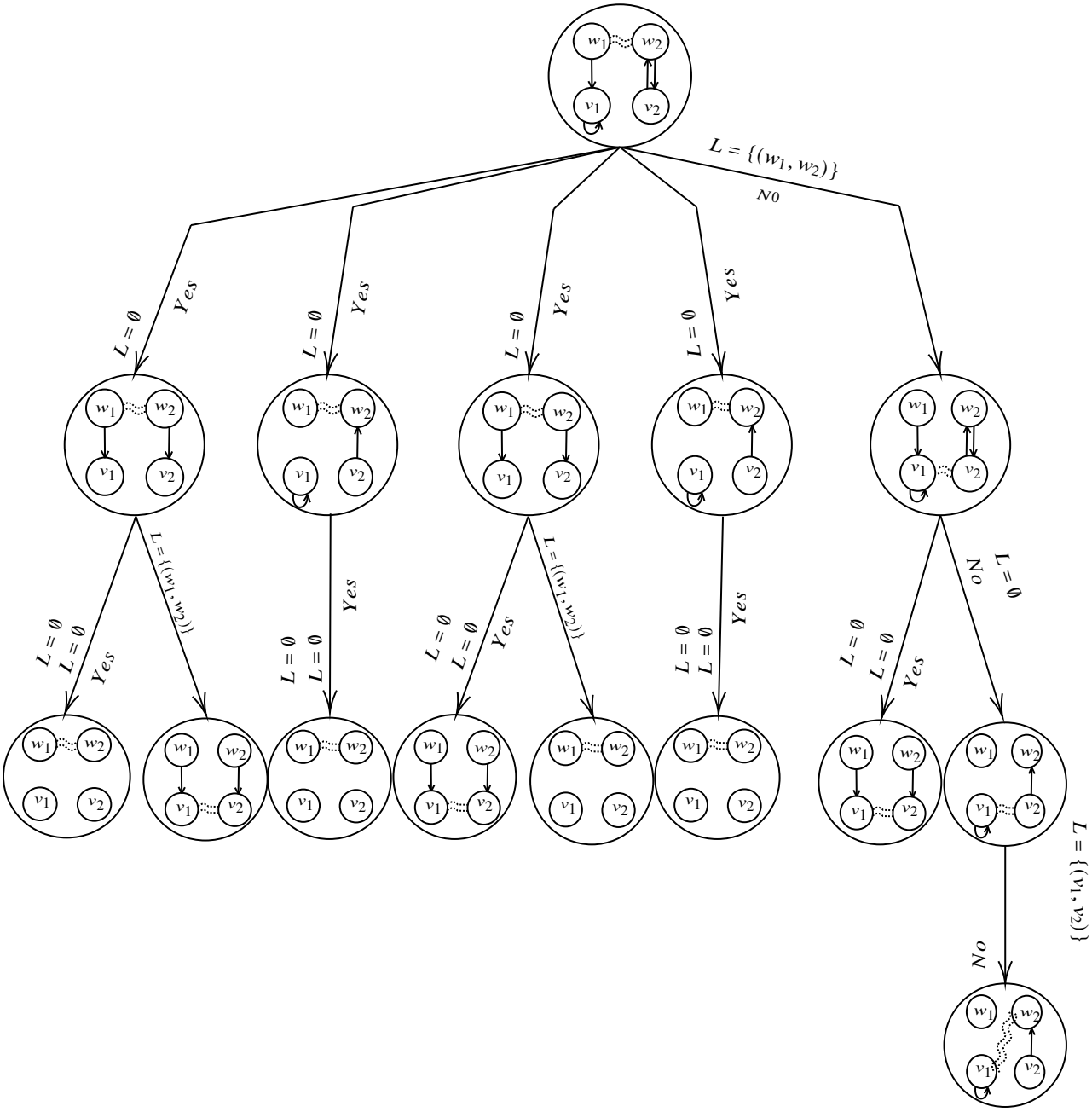
> > > $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$ satisfy conditions (1), (4) and (5) in the definition of $\langle gsb \rangle-$bisimilarity.

> > > For every n, $\exists v_1 \in W_1$ such that $w_1 R_1^n v_1$ iff $\exists v_2 \in W_2$ such that $w_2 R_2^n v_2$; and $(\mathcal{M}_1, v_1)$ and $(\mathcal{M}_2, v_2)$ satisfy condition (1), (4) and (5) from the definition of $\langle gsb \rangle-$bisimilarity. But these conditions are same as the conditions in definition of $\langle gsb \rangle-$bisimilarity. Hence, $(\mathcal{M}_1, w_1) \underleftrightarrow{\leftrightarrow}_s (\mathcal{M}_2, w_2)$ and the function does not return No in this this case.

This completes the proof. □

3.2.1 An example

Below is an example run of the algorithm for gen-Bisimilar for $state = \langle gsb \rangle - bisimilar$. Proposition p is true in all the worlds of both the models. The recursion graph shows all the important nodes. The diagram shows calls to function "checkGeneralizedEdgeDeletion".

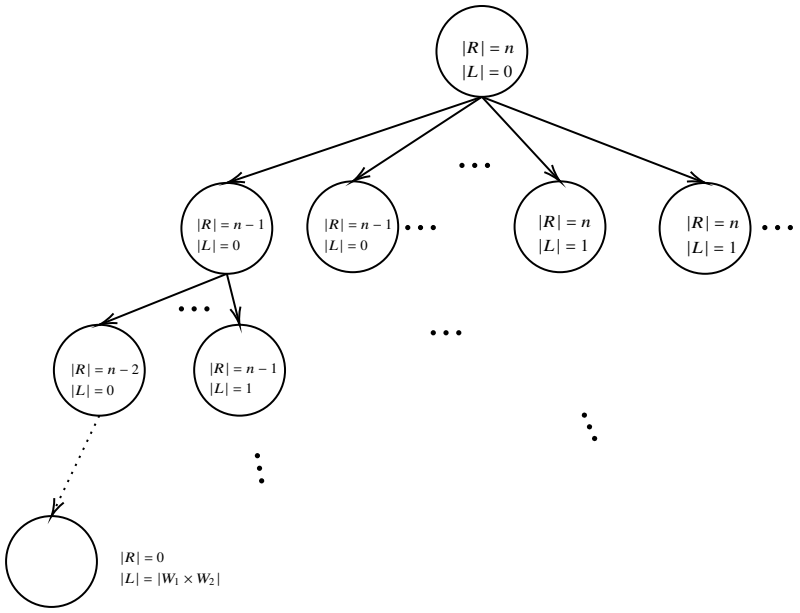


In the above run of the algorithm, not all children of the root returns 'Yes', and hence the input models are not $\langle gsb \rangle - bisimilar$.

3.2.2 On complexity

Theorem 4. *Function* gen-Bisimilar *terminates and is in PSPACE for* $state = \langle gsb \rangle - bisimilar$.

Proof. We will form a recursion tree to see whether the function gen-Bisimilar terminates for $state = \langle gsb \rangle - bisimilar$ and analyze the space complexity of the function.



- 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 checkGeneralizedEdgeDeletion is called. The number of edges in the models for each successive call to checkGeneralizedEdgeDeletion is strictly less than n (namely, $n-1$). Another algorithm that is called is checkSuccessors. In this algorithm, $|L|$ strictly increases. 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 gen-Bisimilar. 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 the 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 *states*

The main difference in the run of the algorithm 1, based on the different values of $state$, is that it calls different functions, namely, $checkEdgeDeletion$ in case of $\langle sb \rangle$-bisimilar, $checkNodeDeletion$ in case of $\langle de \rangle$-bisimilar, and so on. These functions check the analogous conditions (4) and (5) for different notions of bisimilarity. The rest of the algorithm remains same. Therefore the correctness proofs for other notions of bisimilarity are very similar to that of the case of $state = \langle gsb \rangle - bisimilar$. And, the complexity for all bisimilarity problems remains to be in PSPACE. So, we have the following main theorem of this work.

Theorem 5. *Given two pointed relational models, $(\mathcal{M}_1, w_1)$ and $(\mathcal{M}_2, w_2)$, the problem to decide whether they are bisimilar in any of the eight notions described in section 2.2, is in PSPACE.*

4 Further remarks

Till now, we have presented several existing logics concerning model-changing and studied the notion of model comparison or bisimulation for these logics from the algorithmic point of view, and in process we have also shed some

light into the complexity of these problems. We now provide some discussions on lower bounds of these complexity problems which are the next natural questions to answer.

The complexity for checking whether given two pointed models are bisimilar, in basic modal logic, is known to be in polynomial time [38]. What exactly makes the problem of up-bisimilarity more complex (strictly more complex if PTIME is different than PSPACE)? The additional conditions (4) and (5) in the definition of up-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 sequences 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 up-bisimilarity on top of the models being bisimilar in the sense of basic modal logic. These conditions are what makes this problem of up-bisimilarity more complex. If we can show that every such function that satisfies the conditions for up-bisimilarity is generated by a function $g : C_1 \to C_2$, then we believe that the complexity of up-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 number of elements bounded by a polynomial in the size of the input models) candidate generator of the relation up-bisimilar, it may be efficient to check whether such a candidate can be extended to a full up-bisimilar relation.

References

[1] Luca Aceto, Anna Ingólfsdóttir, and Jirí Srba. The algorithmics of bisimilarity. In Davide Sangiorgi and Jan J. M. M. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*, volume 52 of *Cambridge tracts in theoretical computer science*, pages 100–172. Cambridge University Press, 2012.

[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*, WoLLIC 2008, page 56–68, Berlin, Heidelberg, 2008. Springer.

[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 Luís Balcázar, Joaquim Gabarró, and Miklos Santha. Deciding Bisimilarity is P-Complete. *Formal Aspects Comput.*, 4:638–648, 1992.

[7] Alexandru Baltag, Zoé Christoff, Rasmus Kræmmer Rendsvig, and Sonja Smets. Dynamic epistemic logics of diffusion and prediction in social networks. *Studia Logica*, 107, 07 2018.

[8] Alexandru Baltag, Dazhu Li, and Mina Young Pedersen. On the right path: A modal logic for supervised learning. In Patrick Blackburn, Emiliano Lorini, and Meiyun Guo, editors, *Logic, Rationality, and Interaction*, pages 1–14, Berlin, Heidelberg, 2019. Springer Berlin Heidelberg.

[9] Alexandru Baltag and Lawrence S. Moss. Logics for epistemic programs. *Synthese*, 139:165–224, 2004.

[10] Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. The logic of public announcements and common knowledge and private suspicions. In *TARK*, 1998.

[11] Zoé Christoff. *Dynamic Logics of Networks: Information Flow and the Spread of Opinion*. PhD thesis, University of Amsterdam, 2016.

[12] Zoé Christoff and Jens Ulrik Hansen. A logic for diffusion in social networks. *Journal of Applied Logic*, 13(1):48–77, 2015.

[13] 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*, 15:36 – 72, 1993.

[14] Pierre Duchet and Henry Meyniel. Kernels in directed graphs: a poison game. *Discrete Mathematics*, 115(1-3):273–276, 1993.

[15] Raul Fervari. Relation-changing modal logics. *PhD diss., Facultad de Matemática Astronomía y Física, Universidad Nacional de Córdoba, Argentina*, 2014.

[16] Aviezri S. Fraenkel and Edward R. Scheinerman. A deletion game on hypergraphs. *Discrete Applied Mathematics*, 30:155–162, 1991.

[17] Dov M. Gabbay. *Reactive Kripke Semantics*. Springer, 2013.

[18] Hubert Garavel, Frédéric Lang, Radu Mateescu, Gwen Salaün, and Wendelin Serwe. CADP: A Toolbox for the Construction and Analysis of Distributed Processes. In *World Congress on Formal Methods*, 2012.

[19] Jelle Gerbrandy and Willem Groeneveld. Reasoning about information change. *Journal of Logic, Language and Information*, 6:147–169, 1997.

[20] Erich Grädel. Back and forth between logic and games. In Krzysztof R. Apt and Erich Grädel, editors, *Lectures in Game Theory for Computer Scientists*, pages 99–145. Cambridge University Press, 2011.

[21] Jan Friso Groote, Jeroen Keiren, Aad Mathijssen, Bas Ploeger, Frank Stappers, Carst Tankink, Yaroslav Usenko, Muck van Weerdenburg, Wieger Wesselink, Tim Willemse, et al. The mCRL2 toolset. In *Proceedings of the International Workshop on Advanced Software Development Tools and Techniques (WASDeTT 2008)*, page 53, 2008.

[22] 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*, pages 1994–1996, 2019.

[23] Sten Grüner, Frank G. Radmacher, and Wolfgang Thomas. Connectivity games over dynamic networks. *Theoretical Computer Science*, 493:46–65, 2013.

[24] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *Sigact News*, 32(1):60–65, 2001.

[25] 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, New York,USA, 1983. Association for Computing Machinery.

[26] Barteld Kooi. Expressivity and completeness for public update logics via reduction axioms. *Journal of Applied Non-Classical Logics*, 17:231 – 253, 2007.

[27] Barteld Kooi and Bryan Renne. Arrow update logic. *The Review of Symbolic Logic*, 4(4):536–559, 2011.

[28] Barteld Kooi and Bryan Renne. Generalized arrow update logic. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK XIII, page 205–211, New York, 2011.

[29] Dmitriy Kvasov. On sabotage games. *Operation Research Letters*, 44(2):250–254, 2016.

[30] Dazhu Li. Losing connection: the modal logic of definable link deletion. *Journal of Logic and Computation*, 30(3):715–743, 04 2020.

[31] Fenrong Liu, Jeremy Seligman, and Patrick Girard. Logical dynamics of belief change in the community. *Synthese*, 191(11):2403–2431, 2014.

[32] Christof Löding and Philipp Rohde. Model checking and satisfiability for sabotage modal logic. In Pandya Paritosh and Jaikumar Radhakrishnan, editors, *Foundations of Software Technology and Theoretical Computer Science. FSTTCS 2003*, number 2914 in Lecture Notes in Computer Science, pages 302–313. Springer Berlin Heidelberg, 2003.

[33] Christof Löding and Philipp Rohde. Solving the sabotage game is pspace-hard. In Branislav Rovan and Peter Vojtáš, editors, *Mathematical Foundations of Computer Science 2003*, number 2914 in Lecture Notes in Computer Science, pages 531–540. Springer Berlin Heidelberg, 2003.

[34] Sergio Fernando Mera. *Modal memory logics*. PhD thesis, Université Henri Poincaré-Nancy 1, 2009.

[35] Richard Nowakowski and Paul Ottaway. Vertex deletion games with parity rules. *Integers: Electronic Journal of Combinatorial Number Theory*, 5(2), 2005.

[36] Richard Nowakowski and Peter Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.

[37] Cormac O'Sullivan. A vertex and edge deletion game on graphs. *ArXiv*, abs/1709.01354, 2018.

[38] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[39] Jan A. Plaza. Logics of public communications. In M. L. Emrich, M. S. Pfeifer, M. Hadzikadic, and Z. W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems (ISMIS 1989)*, Poster Session Program, page 201–216, Charlotte, North Carolina, 1989. Oak Ridge National Laboratory, ORNL/DSRD-24.

[40] Jan A. Plaza. Logics of public communications. *Synthese*, 158:165–179, 2007.

[41] Gerard R Renardel De Lavalette. Changing modalities. *Journal of Logic and Computation*, 14(2):251–275, 2004.

[42] Philipp Rohde. *On games and logics over dynamically changing structures*. PhD thesis, Aachen, Techn. Hochsch., Diss., 2005.

[43] Declan Thompson. Local fact change logic. In Fenrong Liu, Hiroakira Ono, and Junhua Yu, editors, *Knowledge, Proof and Dynamics*, pages 73–96, Singapore, 2020.

[44] Johan van Benthem. Dynamic odds and ends. Technical report, Technical Report ML-1998-08, University of Amsterdam, 1998.

[45] Johan van Benthem. An essay on sabotage and obstruction. In Dieter Hutter and Werner Stephan, editors, *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, pages 268–276. Springer, Heidelberg, 2005.

[46] Johan van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17:129–155, 01 2007.

[47] Johan van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2011.

[48] Johan van Benthem, Lei Li, Chenwei Shi, and Haoxuan Yin. Hybrid sabotage modal logic. *Journal of Logic and Computation*, 03 2022. exac006.

[49] Johan van Benthem and Fenrong Liu. Dynamic logic of preference upgrade. *Journal of Applied Non-Classical Logics*, 17:157–182, 01 2007.

[50] Johan van Benthem and Fenrong Liu. Graph games and logic design. In Fenrong Liu, Hiroakira Ono, and Junhua Yu, editors, *Knowledge, Proof and Dynamics*, pages 125–146. Springer, Singapore, 2020b.

[51] Johan van Benthem, Krzysztof Mierzewski, and Francesca Zaffora Blando. The modal logic of stepwise removal. *The Review of Symbolic Logic*, page 1–28, 2020.

[52] Johan van Benthem, Jan van Eijck, and Barteld P. Kooi. Logics of communication and change. *Inf. Comput.*, 204:1620–1662, 2006.

[53] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. Dynamic epistemic logic with assignment. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 141–148, 2005.

[54] Hans Van Ditmarsch, Wiebe van Der Hoek, and Barteld Kooi. *Dynamic epistemic logic*, volume 337. Springer Science & Business Media, 2007.

[55] Hans van Ditmarsch, Wiebe van der Hoek, Barteld Kooi, and Louwe B. Kuijer. Arbitrary arrow update logic. *Artificial Intelligence*, 242:80–106, 2017.

[56] Francesca Zaffora Blando, Krzysztof Mierzewski, and Carlos Areces. The modal logics of the poison game. In *Knowledge, Proofs and Dynamics*. Springer, March 2020.

[57] Tianwei Zhang. Solution complexity of local variants of sabotage game. In Fenrong Liu, Hiroakira Ono, and Junhua Yu, editors, *Knowledge, Proof and Dynamics*, pages 3–23, Singapore, 2020.