REPORTS ON MATHEMATICAL LOGIC
48 (2013), 101–115
DOI: 10.4467/20842589RM.13.005.1257

Tin PERKOV

TABLEAU-BASED BISIMULATION INVARIANCE TESTING

A b s t r a c t. A tableau procedure that tests bisimulation invariance of a given first-order formula, and therefore tests if that formula is equivalent to the standard translation of some modal formula, is presented. The test is sound and complete: a given formula is bisimulation invariant if and only if there is a closed tableau for that formula. The test generally does not terminate, but it does if a given formula is bisimulation invariant, i.e., the test is positive.

1. Introduction

In the semantics of Kripke models, modal logic can be regarded as a fragment of first-order logic, since the definition of truth of a modal formula is expressible in the appropriate first-order language. It is more involved to consider this correspondence in the opposite direction, by exploring those

Received 17 April 2012

first-order formulas that are in essence modal. To keep the notation simple, only the *basic modal language* (BML) is considered here, but all the results are easily generalized to the multi-modal context. The alphabet of BML consists of countably many propositional letters which we denote by p, q, and so on, propositional constants $\top$ and $\bot$, Boolean connectives $\neg$, $\vee$, $\wedge$, and modal operators $\Diamond$ and $\Box$. The syntax of modal formulas is given by

$$\varphi ::= p \,|\, \bot \,|\, \top \,|\, \neg\varphi \,|\, \varphi_1 \vee \varphi_2 \,|\, \varphi_1 \wedge \varphi_2 \,|\, \Diamond\varphi \,|\, \Box\varphi,$$

where p ranges over the set of propositional letters. Some of the symbols of the alphabet could have been left out and defined as abbreviations as usual, but it will be more convenient in what follows to take these symbols as basic. We often use $\varphi \to \psi$ instead of $\neg\varphi \vee \psi$.

A *Kripke model* for BML is $\mathfrak{M} = (W, R, V)$, where W is a non-empty set, R is a binary relation on W, and V is the *valuation*, a function that maps every propositional letter p to a subset $V(p) \subseteq W$.

Let σ denote the first-order vocabulary which consists of one binary relation symbol R and a unary relation symbol P for every propositional letter p. Clearly, a Kripke model can be considered a σ-structure: its universe is $|\mathfrak{M}| = W$, and the interpretations of relation symbols are $R^{\mathfrak{M}} = R$ and $P^{\mathfrak{M}} = V(p)$ for every p. Modal formulas are translated to this first-order language by the *standard translation*, a mapping defined as follows:

$$\begin{aligned}
ST_x(p) &= Px, \text{ for each propositional letter } p\\
ST_x(\bot) &= \bot\\
ST_x(\top) &= \top\\
ST_x(\neg\varphi) &= \neg ST_x(\varphi)\\
ST_x(\varphi \vee \psi) &= ST_x(\varphi) \vee ST_x(\psi)\\
ST_x(\varphi \wedge \psi) &= ST_x(\varphi) \wedge ST_x(\psi)\\
ST_x(\Diamond\varphi) &= \exists y(Rxy \wedge ST_y(\varphi))\\
ST_x(\Box\varphi) &= \forall y(Rxy \to ST_y(\varphi)),
\end{aligned}$$

where y in the last two clauses is a fresh variable. The Kripke semantics for modal logic is usually defined in the metalanguage, which is omitted here, but the semantics should be clear from the fact that a modal formula

φ is true in $w \in W$, which is denoted by $\mathfrak{M}, w \Vdash \varphi$, if and only if $\mathfrak{M} \models ST_x(\varphi)[w]$, i.e., if and only if the standard translation of φ is true in $\mathfrak{M}$ under assignment of w to the variable x. As the standard translation is actually a formally rewritten definition of the truth of a modal formula, this fact is easily proved (cf. [1] for details).

So, every modal formula has a first-order equivalent (with one free variable x) in this sense. The converse does not hold, since modal formulas are bisimulation invariant, which first-order formulas need not be.

A *bisimulation* between models $\mathfrak{M} = (W, R, V)$ and $\mathfrak{M}' = (W', R', V')$ is a non-empty relation $Z \subseteq W \times W'$ such that:

(at) if wZw', then for every p we have $w \in V(p)$ if and only if $w' \in V'(p)$;
(forth) if wZw' and Rwv, then there is v' such that vZv' and $R'w'v'$;
(back) if wZw' and $R'w'v'$, then there is v such that vZv' and Rwv.

We say that a σ-formula $F(x)$ is *bisimulation invariant* if the following holds: if there is a bisimulation Z between $\mathfrak{M}$ and $\mathfrak{M}'$ such that wZw', then we have $\mathfrak{M} \models F(x)[w]$ if and only if $\mathfrak{M}' \models F(x)[w']$.

By the Van Benthem Characterization Theorem, the bisimulation invariance is actually what characterizes the modal fragment of first-order logic: a σ-formula is bisimulation invariant if and only if it is equivalent to the standard translation of some modal formula. In other words, an elementary model property is modally definable if and only if it is bisimulation invariant.

This is a result of great importance (see e.g., [1] for the proof) – it establishes bisimulation invariance as an essentially modal model-theoretic notion. Nevertheless, when it comes to practical applications, there is a problem that van Benthem himself points out in [6]: it is undecidable whether a given first-order formula is bisimulation invariant. This is proved by a simple reduction of the problem of the validity of a first-order sentence, which is well known to be undecidable, to the problem of bisimulation invariance (cf. [6] for details).

Still, we can use Characterization Theorem to prove that a model property is not modally definable, by giving an example that shows that this property is not bisimulation invariant. On the other hand, to show that a property is modally definable, we just need to give a modal formula that defines it. Since it is not always easy to do this by hand, the aim of this paper is to develop a procedure that would do this automatically, even though there cannot be a procedure that would decide bisimulation invariance in all cases. It makes sense to try to do this, as much as it makes sense to consider tests for the first-order validity problem. In fact, these problems are closely related when it comes to decidability – not only does the first-order validity reduce to the bisimulation invariance, but also the bisimulation invariance reduces to the first-order validity.

Actually, the tableau procedure for testing bisimulation invariance that is presented in this paper is based on the first-order tableau (or FO-tableau) procedure (see e.g., [5] for the reference). A tableau is in essence a systematic search for a model that satisfies a given formula, so the validity of a formula is tested by a tableau for its negation. The idea of bisimulation invariance testing is also to search for a counterexample, but in this case it means to construct two models and a bisimulation between them that does not preserve the truth of a given formula. To be more precise, for a first-order formula $F(x)$, the procedure tries to build $\mathfrak{M}$ and $\mathfrak{M}'$ and a bisimulation between them such that wZw' and $\mathfrak{M} \models F(x)[w]$ but $\mathfrak{M}' \not\models F(x)[w']$.

In what follows it is shown that the bisimulation invariance tableau simply reduces to the usual FO-tableau and has the same important properties: although the problem is undecidable, the tableau procedure is sound and complete. The procedure terminates in case of a bisimulation invariant formula. In case of a formula that is not bisimulation invariant, the procedure might not terminate, because in some cases the only counterexamples are infinite. However, with an adjustment that is presented in Section 3, it will terminate if there exists a finite example. As usual, a counterexample can be read off an open branch.

2. Rules, soundness and completeness

Let $F(x)$ be a σ-formula in which only variable x is free. A *bisimulation invariance tableau* or a *BI-tableau* for $F(x)$ is a tree obtained in the way that is described in the following. Let U and U' be disjoint sets of constant symbols, and let $Z \notin \sigma$ be a binary relation symbol. Each node of a BI-tableau is a triple (A, B, C), where A is a $\sigma \cup U$-formula or the empty word ε, C is a $\sigma \cup U'$-formula or $C = \varepsilon$, and B is an atomic $\{Z\} \cup U \cup U'$-formula of the form aZc such that $a \in U$ and $c \in U'$, or $B = \varepsilon$. Instead of (A, B, C), nodes will be denoted by $A \cdot B \cdot C$. In this notation, the empty word is not denoted, so for example a node such that $B = C = \varepsilon$ is denoted by $A \cdot \cdot$ and so on.

Let A be any first-order formula. Denote by $A(c/x)$ a formula obtained from A by substituting every free occurrence of a variable x with a constant symbol c. The root of a BI-tableau for $F(x)$ is

$$F(w/x) \cdot wZw' \cdot \neg F(w'/x)$$

To reduce the number of rules and to simplify proofs, we assume that both $F(w/x)$ and $\neg F(w'/x)$ are in the negation normal form (NNF), i.e., rewritten (if necessary) as an equivalent formula in which only atomic subformulas can be in the scope of negation, while $\neg$, $\vee$ and $\wedge$ are the only allowed Boolean connectives.

Formulas at the root suggest that by applying some rules we will try to satisfy F at w and $\neg F$ at w' by building $\mathfrak{M}$ and $\mathfrak{M}'$ starting from these initial elements, together with a bisimulation Z between them such that wZw'. So, formulas on the left-hand side of any node of a BI-tableau concern $\mathfrak{M}$, on the right-hand side $\mathfrak{M}'$, and formulas in the middle concern the (potential) bisimulation between them.

Each node of a BI-tableau is obtained from some formulas of its ancestors by applying one of the following rules. There are two groups of rules. The first are standard FO-tableau rules, which apply either to some formula on the left or on the right side. Each rule has the left side and the right side version.

- $\vee$-*rule*

$$\begin{array}{c} A_1 \vee A_2 \cdot B \cdot C \\ \swarrow \quad \searrow \\ A_1 \cdot \cdot \qquad A_2 \cdot \cdot \end{array} \qquad\qquad \begin{array}{c} A \cdot B \cdot C_1 \vee C_2 \\ \swarrow \quad \searrow \\ \cdot \cdot C_1 \qquad \cdot \cdot C_2 \end{array}$$

- $\wedge$-*rule*

$$\begin{array}{c} A_1 \wedge A_2 \cdot B \cdot C \\ A_1 \cdot \cdot \\ A_2 \cdot \cdot \end{array} \qquad\qquad \begin{array}{c} A \cdot B \cdot C_1 \wedge C_2 \\ \cdot \cdot C_1 \\ \cdot \cdot C_2 \end{array}$$

- $\exists$-*rule*

$$\begin{array}{lcl} \exists xA \cdot B \cdot C & \qquad & A \cdot B \cdot \exists xC \\ A(a/x) \cdot \cdot & & \cdot \cdot C(a'/x), \end{array}$$

  where a (resp. a') is a new constant symbol, i.e., it does not occur at any ancestor node.

- $\forall$-*rule*

$$\begin{array}{lcl} \forall xA \cdot B \cdot C & \qquad & A \cdot B \cdot \forall xC \\ A(a/x) \cdot \cdot & & \cdot \cdot C(a'/x), \end{array}$$

  where a (resp. a') is any constant symbol that occurs on the left (resp. right) side of any ancestor or descendant node.

As usual, each of these rules is applied only once to each appropriate node, except for the $\forall$-rule, which is applied once for each constant symbol that occurs on the appropriate side of any node in a tableau.

The second group of rules are the bisimulation rules. These involve both sides and the middle of a node, and apply only to atomic formulas. Unlike the first-order rules, each of the bisimulation rules uses two of the ancestor nodes to obtain a new node (except in the case when the root contains both premises – see Example 5). These rules are applied only once to each appropriate pair of nodes. So, different applications may share one premise, but not both.

- (forth)-*rule*

$$\begin{array}{l} Rab \cdot \cdot \\ A \cdot aZa' \cdot C \\ \quad \cdot\, bZb' \cdot \boxed{Ra'b'} \end{array}$$

(where b' is new)

- (back)-*rule*

$$\begin{array}{r} \cdot \cdot Ra'b' \\ A \cdot aZa' \cdot C \\ \boxed{Rab} \cdot bZb' \cdot \quad \end{array}$$

(where b is new)

- (at)-*rule*

$$\begin{array}{ccc} Pa\cdot\cdot & \qquad\qquad & \cdot\cdot Pa' \\ A\cdot aZa'\cdot C & & A\cdot aZa'\cdot C \\ \cdot\cdot\boxed{Pa'} & & \boxed{Pa}\cdot\cdot \end{array}$$

These rules clearly resemble (forth), (back) and (at) conditions from the definition of bisimulation. Atomic formulas appended by all of these rules are depicted boxed. Bisimulation rules do not use nodes with boxed formulas as premises.

We say that a formula (or a pair of formulas in the case of bisimulation rules) in a tableau is *used* if the appropriate rule is applied. In case of ∀-rule, this means that it is applied for each constant symbol that occurs on the appropriate side of any node.

A branch of a BI-tableau is *closed* if some formula and its negation occur at some nodes of that branch. In examples, closed branches will be terminated by a symbol X. A BI-tableau is *closed* if all of its branches are closed. A branch (a BI-tableau) is *open* if it is not closed. A branch is *completed* if it is closed or infinite or it cannot be further extended, i.e., all non-atomic formulas and all appropriate pairs of atomic formulas have been used. A tableau is *completed* if all of its branches are completed.

Before turning to general arguments which show that the procedure is sound and complete, consider several examples.

Example 1. The following is a BI-tableau for the formula $\exists yRxy$:

$$\begin{array}{ll} \exists yRwy\cdot wZw'\cdot\forall y\neg Rw'y & \\ \quad Rwa\cdot\cdot & (\exists) \\ \qquad\cdot aZa'\cdot\boxed{Rw'a'} & \text{(forth)} \\ \qquad\cdot\cdot\neg Rw'a' & (\forall) \\ \qquad X & \end{array}$$

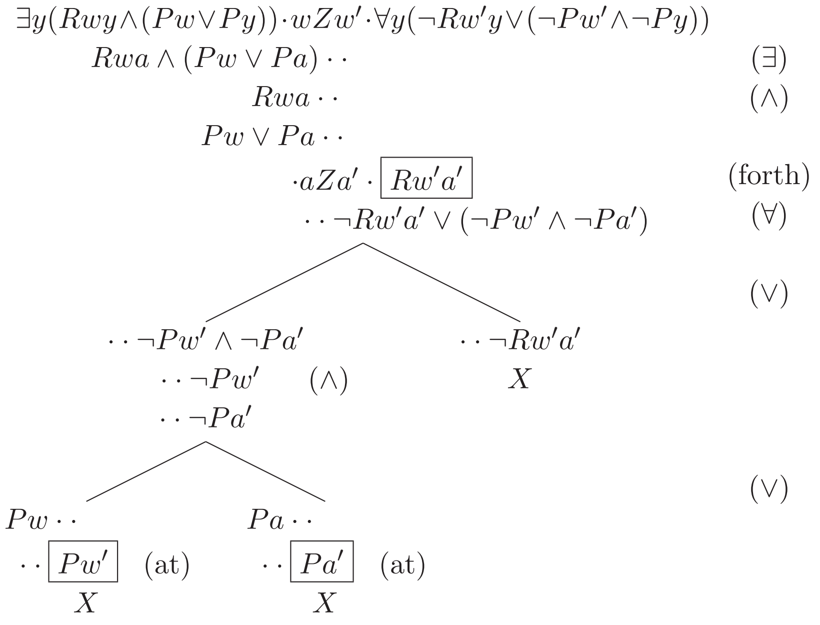
First the ∃-rule on the left side introduced a, then the (forth)-rule is applied to introduce a' on the right side, and finally, ∀-rule is applied on the right side for a'. Note that ∀-rule would also apply for w', but the branch is already closed, so there is no need for further application of rules. Since there is no branching, the tableau is closed. This means that we have found a contradiction while trying to construct a counterexample, so the procedure gives the answer that the initial formula is bisimulation invariant. Indeed, it is equivalent to the standard translation of $\Diamond\top$.

Example 2. $(F(x) = Px \lor \exists yRyy)$

$$\begin{tikzcd}[arrows=-]
 & Pw \lor \exists yRyy \cdot wZw' \cdot \neg Pw' \land \forall y\neg Ryy & \\
 & \cdot\cdot \neg Pw' & (\land) \\
 & \cdot\cdot \forall y\neg Ryy \arrow[dl] \arrow[dr] & (\lor) \\
Pw \cdot\cdot & & \exists yRyy \cdot\cdot \\
\cdot\cdot \boxed{Pw'} \quad (\text{at}) & & Raa \cdot\cdot \quad (\exists) \\
X & & \cdot\cdot \neg Rw'w' \quad (\forall)
\end{tikzcd}$$

The left branch is closed, but the right one is open and completed. This means that there is a counterexample that shows that $F(x)$ is not bisimulation invariant. This example is clearly read off the open branch: the model read off from the left side is $\mathfrak{M}$, with universe $|\mathfrak{M}| = \{w, a\}$ and interpretations $R^{\mathfrak{M}} = \{(a, a)\}$ and $P^{\mathfrak{M}} = \emptyset$. From the right side we have $\mathfrak{M}'$ with $|\mathfrak{M}'| = \{w'\}$, $R^{\mathfrak{M}'} = \emptyset$ and $P^{\mathfrak{M}'} = \emptyset$. All relation symbols that do not occur in formulas of the tableau are interpreted in both models by the empty set. A bisimulation between them is $Z = \{(w, w')\}$. Indeed, it is easy to see that $\mathfrak{M} \models F(x)[w]$ and $\mathfrak{M}' \models \neg F(x)[w']$, and that Z is really a bisimulation.

Example 3. $(F(x) = \exists y(Rxy \wedge (Px \vee Py)))$


$\exists y(Rwy \wedge (Pw \vee Py)) \cdot wZw' \cdot \forall y(\neg Rw'y \vee (\neg Pw' \wedge \neg Py))$
$Rwa \wedge (Pw \vee Pa) \cdot\cdot$ $(\exists)$
$Rwa \cdot\cdot$ $(\wedge)$
$Pw \vee Pa \cdot\cdot$
$\cdot aZa' \cdot \boxed{Rw'a'}$ (forth)
$\cdot\cdot \neg Rw'a' \vee (\neg Pw' \wedge \neg Pa')$ $(\forall)$
$(\vee)$
$\cdot\cdot \neg Pw' \wedge \neg Pa'$
$\cdot\cdot \neg Rw'a'$
$\cdot\cdot \neg Pw'$ $(\wedge)$
X
$\cdot\cdot \neg Pa'$
$(\vee)$
$Pw \cdot\cdot$
$Pa \cdot\cdot$
$\cdot\cdot \boxed{Pw'}$ (at)
$\cdot\cdot \boxed{Pa'}$ (at)
X
X

The tableau is closed. Indeed, it is easy to see that $F(x)$ is equivalent to the standard translation of $(\Diamond\top \wedge p) \vee \Diamond p$.

Example 4. Let $F(x) = \exists y(Rxy \wedge \neg Rxy)$. This is clearly an unsatisfiable formula. Consider the tableau.

$$\begin{array}{cr}
\exists y(Rwy \wedge \neg Rwy) \cdot wZw' \cdot \forall y(R'w'y \vee \neg R'w'y) & \\
Rwa \wedge \neg Rwa \cdot\cdot & (\exists) \\
Rwa \cdot\cdot & (\wedge) \\
\neg Rwa \cdot\cdot & \\
X &
\end{array}$$

Rules are applied on the left side only, so this is actually an ordinary FO-tableau that shows the unsatisfiability of $F(x)$. The tableau is closed, and rightly so, because any unsatisfiable formula is equivalent to $\bot$. Similarly, any valid formula is equivalent to $\top$.

The following notion will be needed in the proof of soundness and completeness. Let F be a σ-formula and let $P \notin \sigma$ be a unary relation symbol.

The *P-relativization* of F is $\sigma \cup P$-formula F^P defined inductively as follows: $F^P = F$ in the atomic case, the relativization is compositional with all Boolean connectives, and for quantifiers we have: if $F = \forall x A$ then $F^P = \forall x(Px \to A^P)$, if $F = \exists x A$ then $F^P = \exists x(Px \land A^P)$.

Let $\mathfrak{M}$ be a $\sigma \cup P$-structure such that $P^{\mathfrak{M}}$ is a σ-closed set (if a language has no function symbols, this simply means that $P^{\mathfrak{M}}$ is non-empty and contains all interpretations of constant symbols) and let $\mathfrak{M}_P$ be the submodel of $\mathfrak{M}$ with the universe $|\mathfrak{M}_P| = P^{\mathfrak{M}}$. Then, by the Relativization Lemma, for any σ-sentence F we have that $\mathfrak{M} \models F^P$ if and only if $\mathfrak{M}_P \models F$ (see e.g., [3] for the proof of the lemma).

Theorem. *The bisimulation invariance tableau calculus is sound and complete: a σ-formula $F(x)$ is bisimulation invariant if and only if there is a closed BI-tableau for $F(x)$.*

Proof. The BI-tableau reduces to the standard FO-tableau as follows. We actually seek a model which is the disjoint union of a model for $F(x)$ and a model for $\neg F(x)$, with a new relation Z between these two models such that (at), (forth) and (back) are satisfied. The key in the reduction is that these conditions can be phrased as first-order formulas. The only danger is that (at) quantifies over unary relation symbols, but we actually need only finitely many of them – those that occur in $F(x)$.

Put $\sigma' = \sigma \cup \{Z, L, D\}$, where $Z \notin \sigma$ is a binary relation symbol and $L, D \notin \sigma$ are unary relation symbols, representing the left and the right side of a BI-tableau. Now consider (the conjunction of) the following sentences:

(1) $\exists x \exists x'(Lx \land Dx' \land F^L(x) \land \neg F^D(x') \land Zxx')$

(2) $\forall y \forall y'(Zyy' \to (Py \leftrightarrow Py'))$, for every $P \in \sigma$ that occurs in $F(x)$

(3) $\forall y \forall y' \forall z((Zyy' \land Ryz) \to \exists z'(Dz' \land Zzz' \land Ry'z'))$

(4) $\forall y \forall y' \forall z'((Zyy' \land Ry'z') \to \exists z(Lz \land Zzz' \land Ryz))$

The sentence (1) represents the root of a BI-tableau, and (2)-(4) represent (at), (forth) and (back), respectively. Assume for the moment that the following claim holds:

($*$) If $F(x)$ is not bisimulation invariant, then there is a model for (1)-(4).

Now, to prove the soundness, assume we have a closed BI-tableau for $F(x)$. Using this BI-tableau, we prove that there is a closed FO-tableau for (1)-(4) as follows. The root of a BI-tableau for $F(x)$ translates into the application of the first-order $\exists$-rule twice, starting from (1). The first group of BI-tableau rules are actually the FO-tableau rules applied to formulas on the left or on the right side, which translates to the same rules being applied to some formulas of the FO-tableau for (1)-(4). The (at)-rule can be viewed as shorthand for two applications of the $\forall$-rule starting from (2), and (forth) and (back)-rules can be viewed as shorthand for three applications of the $\forall$-rule followed by the $\exists$-rule starting from (3) and (4), respectively. Clearly, the FO-tableau constructed in this way is closed. Due to the soundness of the FO-tableau calculus (cf. [5]), in this way it is proved that there is no model for (1)-(4), hence by $(*)$ we have that $F(x)$ is bisimulation invariant, as desired. So, to prove soundness, it remains to show $(*)$.

Assume $F(x)$ is not bisimulation invariant, so there exist σ-structures $\mathfrak{M}, \mathfrak{M}'$ and a bisimulation Z between them such that wZw' and $\mathfrak{M} \models F(x)[w]$ but $\mathfrak{M}' \models \neg F(x)[w']$. Consider the $\sigma \cup \{Z, L, D\}$-expansion $\mathfrak{N}$ of the disjoint union $\mathfrak{M} \uplus \mathfrak{M}'$ such that $L^{\mathfrak{N}} = |\mathfrak{M}|$, $D^{\mathfrak{N}} = |\mathfrak{M}'|$ and $Z^{\mathfrak{N}} = Z$. Now, the Relativization Lemma implies $\mathfrak{N} \models F^L(x)[w]$ and $\mathfrak{N} \models \neg F^D(x)[w']$. Clearly, $\mathfrak{N}$ is a model for (1)-(4).

For the completeness, let $F(x)$ be a bisimulation invariant formula. Systematically we build a completed (possibly infinite) BI-tableau for $F(x)$ by repeating the following:

- Make the next step of the systematic (see [5]) FO-tableau by applying a first-order rule to a formula on the left side of an appropriate node.
- Apply the (forth)-rule if possible.
- Apply the left version of the (at)-rule if possible.
- Make the next step of the systematic FO-tableau for the right side.
- Apply the (back)-rule if possible.
- Apply the right version of the (at)-rule if possible.

Now, assume that there is an open branch in this systematic tableau. Let S be the set of formulas occurring on the left side of each node of this branch. Since the branch is open, there is no atomic $A \in S$ such that $\neg A \in S$, so since the tableau is systematic, it is easy to see that S is a Hintikka set (cf. [5]). Hence, S is satisfiable in a model $\mathfrak{M}$ such that $|\mathfrak{M}|$ is the set of all constant symbols that occur in formulas from S, so clearly $\mathfrak{M} \models F(x)[w]$. Similarly we conclude that there is $\mathfrak{M}'$ satisfying the right side of the branch such that $\mathfrak{M}' \models \neg F(x)[w']$. Furthermore, since the BI-tableau is completed, implying that bisimulation rules are applied whenever needed, it is easy to see that formulas from the middle component of all nodes of the branch define a bisimulation between $\mathfrak{M}$ and $\mathfrak{M}'$ such that wZw'. But this contradicts the assumption that $F(x)$ is bisimulation invariant. Hence, no branch of a systematic BI-tableau can be open, so this tableau is closed, which implies completeness. □

3. Termination

We say that a completed tableau *terminates* if it is closed or if it is open and finite. For any bisimulation invariant formula, the systematic BI-tableau, built as described in the proof of the completeness theorem, clearly terminates, since it must be closed. Due to the undecidability, obviously some systematic tableaux do not terminate, and any such tableau must be for a formula that is not bisimulation invariant. Notably, any such formula that does not have a finite counterexample does not have a terminating tableau. Examples of this situation can be constructed from known examples of satisfiable first-order formulas with no finite model, via reduction of the first-order validity problem to the bisimulation invariance problem. It remains to examine formulas that are not bisimulation invariant and have a finite counterexample. As Example 3 shows, some of these formulas will have a terminating tableau and if this is the case, then a counterexample can be read off an open branch. Moreover, with an improvement of the rules that introduce new constant symbols, it can be achieved that all such formulas have a terminating tableau. Since this trick is imported from the standard FO-tableau, it will not be considered in full detail here, but just informally described (for more details, and for the proof that this covers all cases with finite counterexamples, see [2] and [4]).

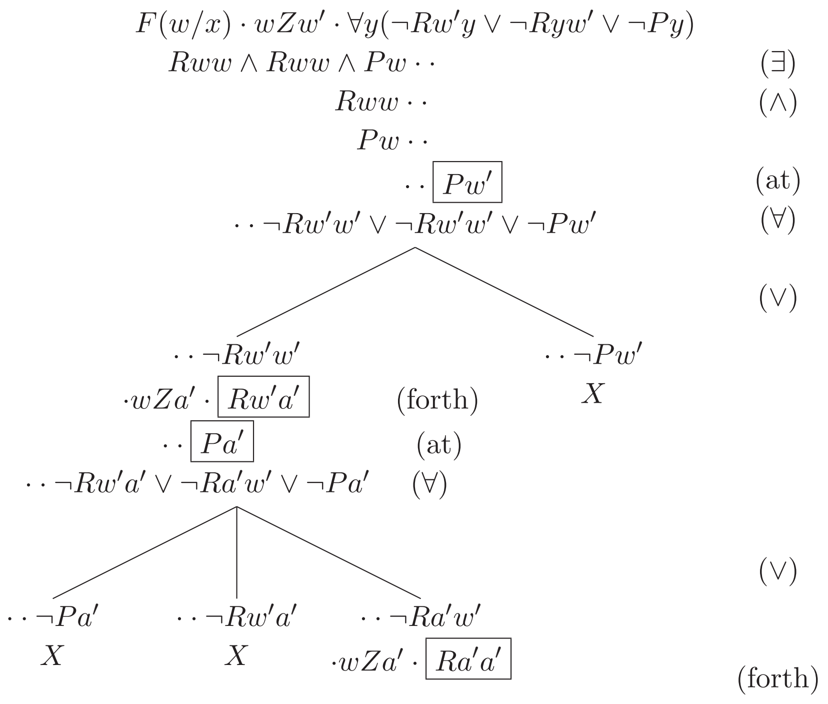
The correction is the following: when applying an introducing rule, first try out the already introduced constant symbols, one by one, until one of them results in a completed open branch. If all of them produce closed branches, introduce a new constant symbol. Consider some examples.

Example 5. $(F(x) = Rxx)$

$$\begin{tikzcd}
 & Rww \cdot wZw' \cdot \neg Rw'w' \arrow[dl, no head] \arrow[dr, no head] & & \\
\cdot wZw' \cdot \boxed{Rw'w'} & & \cdot wZa' \cdot \boxed{Rw'a'} & \text{(forth)} \\
X & & \cdot wZw' \cdot \boxed{Ra'w'} & \text{(forth)}
\end{tikzcd}$$

The root already contains both premises of the (forth)-rule, which should introduce a new constant symbol. Branching follows, which is not due to an application of $\vee$-rule, but it is this try-out branching. On the left branch w' is tried out and results with a closed branch. There is no other already introduced element, so the try-out failed and we need another branch where a new element is introduced as usual. At this point, (forth) is applied again, tried out at w', which leads to a completed open branch, so there is no need to try out further symbols, or to introduce a new constant symbol. The initial formula is not bisimulation invariant, and an example for this is read off the right branch: $|\mathfrak{M}| = \{w\}$, $R^{\mathfrak{M}} = \{(w, w)\}$, $P^{\mathfrak{M}} = \emptyset$ for all unary relation symbols P, and $|\mathfrak{M}'| = \{w', a'\}$, $R^{\mathfrak{M}'} = \{(w', a'), (a', w')\}$, $P^{\mathfrak{M}'} = \emptyset$ for all P. The bisimulation is $Z = \{(w, w'), (w, a')\}$. Note that, without the modification of the introducing rules, the (forth)-rule would force the introduction of new constant symbols infinitely. Also, note that trying out a' (which was not done because w' already resulted in a completed open branch) would produce another counterexample.

Example 6. $(F(x) = \exists y(Rxy \wedge Ryx \wedge Py))$


$F(w/x) \cdot wZw' \cdot \forall y(\neg Rw'y \vee \neg Ryw' \vee \neg Py)$
$Rww \wedge Rww \wedge Pw \cdot\cdot$
$(\exists)$
$Rww \cdot\cdot$
$(\wedge)$
$Pw \cdot\cdot$
$\cdot\cdot \boxed{Pw'}$
(at)
$\cdot\cdot \neg Rw'w' \vee \neg Rw'w' \vee \neg Pw'$
$(\forall)$
$(\vee)$
$\cdot\cdot \neg Rw'w'$
$\cdot\cdot \neg Pw'$
$\cdot wZa' \cdot \boxed{Rw'a'}$
(forth)
X
$\cdot\cdot \boxed{Pa'}$
(at)
$\cdot\cdot \neg Rw'a' \vee \neg Ra'w' \vee \neg Pa'$
$(\forall)$
$(\vee)$
$\cdot\cdot \neg Pa'$
X
$\cdot\cdot \neg Rw'a'$
X
$\cdot\cdot \neg Ra'w'$
$\cdot wZa' \cdot \boxed{Ra'a'}$
(forth)

The $\exists$-rule is tried out at w successfully. At the first application of the (forth)-rule a new constant symbol is introduced. Trying out w' at this point would have resulted in a closed branch, which is not depicted for the sake of readability of the tree. The second application of the (forth)-rule is successfully tried out at a'. Again, trying out w' would have failed.

The counterexample read off the only open branch is: $|\mathfrak{M}| = \{w\}$, $R^{\mathfrak{M}} = \{(w,w)\}$, $P^{\mathfrak{M}} = \{w\}$; $|\mathfrak{M}'| = \{w', a'\}$, $R^{\mathfrak{M}'} = \{(w',a'),(a',a')\}$, $P^{\mathfrak{M}'} = \{w', a'\}$; $Z = \{(w,w'),(w,a')\}$.

4. Conclusion

BI-tableau is in essence an application of FO-tableau calculus to the problem of modal definability of elementary properties of Kripke models. Further research should result in an effective procedure for obtaining a modal formula equivalent to a given FO-formula which is bisimulation invariant.

Hopefully, a closed BI-tableau will be useful as a starting point in building a modal correspondent.

Furthermore, for the purpose of implementation of BI-tableau, we need to make the procedure deterministic. The systematic tableau used in the proof of completeness is probably a fine starting point for this, but note that the examples presented in this paper are not done in this way – steps are chosen with the purpose of terminating faster. Starting from ideas that arise from examples, further work should include developing some general strategies that would make the procedure faster (in cases when it does terminate).

Acknowledgements

I am grateful to Sophia Knight and the anonymous referees for valuable corrections and suggestions.

References

[1] J. Blackburn, M. de Rijke, Y. Venema, Modal Logic, Cambridge University Press, 2001.

[2] G. Boolos, *Trees and Finite Satisfiability: Proof of a Conjecture of Burgess*, Notre Dame Journal of Formal Logic **25** (1984), pp. 193–197.

[3] H.-D. Ebbinghaus, J. Flum, W. Thomas, Mathematical Logic, Springer-Verlag, 1984.

[4] R. Jeffrey, J. P. Burgess, Formal Logic: Its Scope and Limits, Hackett, 2006.

[5] R. M. Smullyan, First-Order Logic, Springer-Verlag, 1968.

[6] J. van Benthem, Exploring Logical Dynamics, Studies in Logic, Language and Information, CSLI Publications & FoLLI, Stanford, 1996.

Polytechnic of Zagreb
Avenija V. Holjevca 15
10000 Zagreb, Croatia
tin.perkov@tvz.hr