Work-In-Progress Papers Presented at The 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2019)

Editors: Serenella Cerrito and Andrei Popescu

Technical Report
Department of Computer Science
Faculty of Science and Technology
Middlesex University, London
ISBN: 9781911371786

Tableau-based translation from first-order logic to modal logic[*]

Tin Perkov

Chair of Mathematics and Statistics, Faculty of Teacher Education, University of Zagreb, Savska c. 77, HR-10000 Zagreb, Croatia

Abstract. We define a procedure for translation of a given first-order formula to an equivalent modal formula, if such exists, by using tableau-based bisimulation invariance test. Previously developed tableau procedure tests bisimulation invariance of a given first-order formula, and therefore tests whether that formula is equivalent to the standard translation of some modal formula. Using a closed tableau as the starting point, we show how an equivalent modal formula can be effectively obtained.

Keywords: Modal logic, Bisimulation invariance, Tableaux.

1 Introduction

Kripke semantics makes modal logic a fragment of first-order logic, namely the bisimulation invariant fragment. Unfortunately, this is an undecidable fragment of first-order logic (see [1]). Truth clauses of modal formulas have the obvious translation to first-order logic. This is the basis of standard translation from modal logic to first-order logic. Of course, the set of these translations is decidable, but the set of all first-order formulas (over the appropriate signature) equivalent to the standard translation of some modal formula is not decidable. By the Van Benthem Characterization Theorem, this is exactly the set of all formulas invariant under bisimulation, which makes the latter the basic equivalence between Kripke models.

In [5] we developed a tableau-based procedure to test whether a given first-order formula is bisimulation invariant. Using reduction to the standard first-order tableau (see e.g. [6] for the reference), we proved soundness and completeness. The latter implies semi-decidability of the problem, since it means that each bisimulation invariant formula has a closed tableau. In other words, in case of the affirmative answer, the procedure does terminate and gives the correct answer, i.e. the test is positive.

The aim of the present work is to use a given closed tableau of a bisimulation invariant formula not only as the answer that this formula is equivalent to the

[*] This work has been supported by Croatian Science Foundation (HRZZ) under the projects UIP-05-2017-9219 and IP-01-2018-7459.

standard translation of some modal formula, but to obtain this modal formula using the tableau as the starting point.

In Section 2 we briefly overview basic notions and results from [5], for the sake of self-containment of the present paper. In Section 3 we present the procedure of obtaining a modal correspondent of a given bisimulation invariant first-order formula. We conclude with a brief further work note.

2 Preliminaries

We assume familiarity with modal logic (see e.g. [2] for further details if needed), so the following several paragraphs are here just in order to fix notation, terminology and a convenient choice of primitive symbols. We will only consider the basic modal language, the alphabet of which extends that of classical propositional logic with mutually dual modalities $\Diamond$ and $\Box$. The syntax of modal formulas is given by

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

where p ranges over the set of propositional variables. We often write $\varphi \to \psi$ instead of $\neg\varphi \vee \psi$.

A *Kripke model* is $\mathfrak{M} = (W, R, V)$, where $W \neq \emptyset$, $R \subseteq W \times W$, and V is the *valuation*, a function that maps each propositional variable p to some $V(p) \subseteq W$.

Let σ be the first-order signature consisting of a binary relational symbol R and a unary relational symbol P for each propositional variable p. A Kripke model can be regarded as a σ-structure, with $|\mathfrak{M}| = W$, $R^{\mathfrak{M}} = R$ and $P^{\mathfrak{M}} = V(p)$ for each p. The *standard translation* is defined as follows:

$$\begin{aligned}
ST_x(p) &= Px, \text{ for each propositional letter } p\\
ST_x(\bot) &= \bot\\
ST_x(\top) &= \top\\
ST_x(\neg\varphi) &= \neg ST_x(\varphi)\\
ST_x(\varphi_1 \vee \varphi_2) &= ST_x(\varphi_1) \vee ST_x(\varphi_2)\\
ST_x(\varphi_1 \wedge \varphi_2) &= ST_x(\varphi_1) \wedge ST_x(\varphi_2)\\
ST_x(\Diamond\varphi) &= \exists y(Rxy \wedge ST_y(\varphi))\\
ST_x(\Box\varphi) &= \forall y(Rxy \to ST_y(\varphi)),
\end{aligned}$$

where y in the last two clauses is a fresh variable. 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.

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, a σ-formula $F(x)$ is bisimulation invariant if and only if it is equivalent to the standard translation of some modal formula.

Generally, a tableau is a systematic search for a model that satisfies a given formula, so we test the validity of a formula by a tableau for its negation. The idea of bisimulation invariance testing is also to search for a counterexample, in this case to construct two models and a bisimulation between them that does not preserve the truth of a given formula $F(x)$.

Let us briefly introduce the rules of *bisimulation invariance tableau* or *BI-tableau.* Let $F(x)$ be a σ-formula in which only variable x is free. Each node of a BI-tableau is a triple (A, B, C), which we write as $A \cdot B \cdot C$ where A and C can be the empty word or a formula over σ expanded with constant symbols introduced in the tableau, while B is of the form cZc', or the empty word. We do not denote the empty word in the tableau, so for example a node such that B and C are empty is denoted $A \cdot \cdot$.

Let A be a 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. only atomic subformulas can be in the scope of negation. Note that this makes writing simply $\neg F(w'/x)$ an abuse of notation, but hopefully without any danger of confusion.

The form of the root suggests that by applying some rules we 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 are about $\mathfrak{M}$, on the right-hand side about $\mathfrak{M}'$, and formulas in the middle are about a relation, possibly a bisimulation, between them.

These are the rules:

– $\vee$-*rule*

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

– $\wedge$-*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*

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

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

– $\forall$-*rule*

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

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

Each of the rules above 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 have two premises each. Distinct applications of each of them 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*

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

(where b is new)

– (at)-*rule*

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

Atomic formulas appended by all of these rules are depicted boxed. These rules do not make further use of nodes with boxed formulas as premises.

We use usual notions for tableaux. Since the present paper focuses on closed tableaux, let us emphasize only that we say that a branch of a BI-tableau is *closed* if some formula and its negation occur at some of its nodes, while a BI-tableau is *closed* if all of its branches are closed.

Theorem 1 ([5]). *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 $F(x)$.*

The procedure terminates in case of a bisimulation invariant formula. Otherwise it might not terminate, since in some cases the only counterexamples are infinite. With an adjustment known from first-order logic tableaux (see [3]), it will always terminate if there exists a finite example. A counterexample can be read off an open branch. In this paper we will not use this adjustment, since we focus on bisimulation invariant formulas and closed tableaux.

3 Obtaining a formula from a closed tableau

The procedure has three stages: first the interim first-order formula is read off the tableau, then it is normalized in a way that enables the final stage – translation to a modal formula.

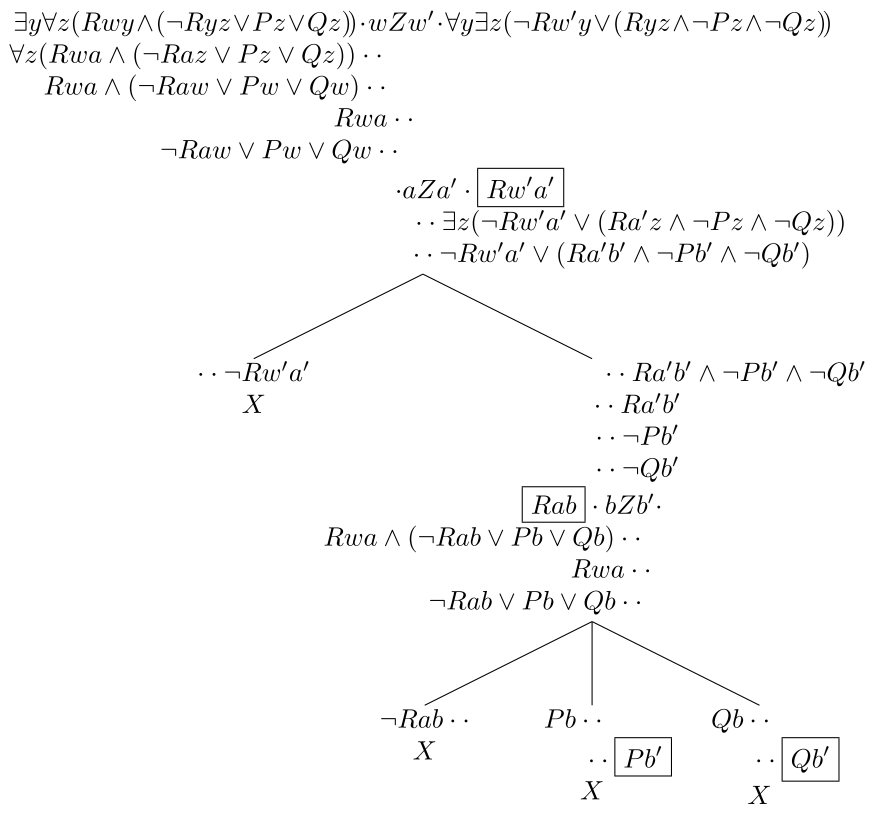
Suppose we have a closed BI-tableau for $F(x)$. Let $a_0 = w$ and let $a_1, a_2, \ldots$ be the other constant symbols that occur in the left-side formulas. We read the tableau starting from the root until one of the following cases occurs and construct an interim first-order formula F' as follows.

– The branch closes. If this is due to an occurence of a boxed formula of the form $\boxed{Pa_i}$ and $\neg Pa_i$ on the left side, put $F' = \neg Px_i$. If such contradiction occurs on the right side, put $F' = Px_i$.
  Similarly, if we have $\boxed{Ra_ia_j}$ and $\neg Ra_ia_j$ on the left, put $F' = \neg Rx_ix_j$, and if we have such contradiction on the right side, put $F' = Rx_ix_j$.
  If the branch closes due to a contradiction of some formulas that are not boxed, put $F' = \bot$ if this contradiction occurs on the left side, but if it occurs on the right side, put $F' = \top$.
– A branching occurs. If this is caused by an application of $\vee$-rule on the left side of the tableau, then put $F' = F_1' \vee F_2'$, or else if it is caused by an application of $\vee$-rule on the right side, then put $F' = F_1' \wedge F_2'$, where F_1' and F_2' are obtained from the respective branches. Proceed inductively.
– An application of (forth)-rule occurs, involving a formula of the form Ra_ja_k on the left-hand side. In this case put $F' = \exists x_k F_1'$ and obtain F_1' from the rest of the current branch inductively.
– An application of (back)-rule occurs, which appends a formula of the form $\boxed{Ra_ja_k}$ on the left-hand side. Put $F' = \forall x_k F_1'$ and obtain F_1' from the rest of the current branch inductively.

In examples with only few constant symbols, to simplify notation, we use constant symbols w, a, b and variables x, y, z.

Consider some examples.

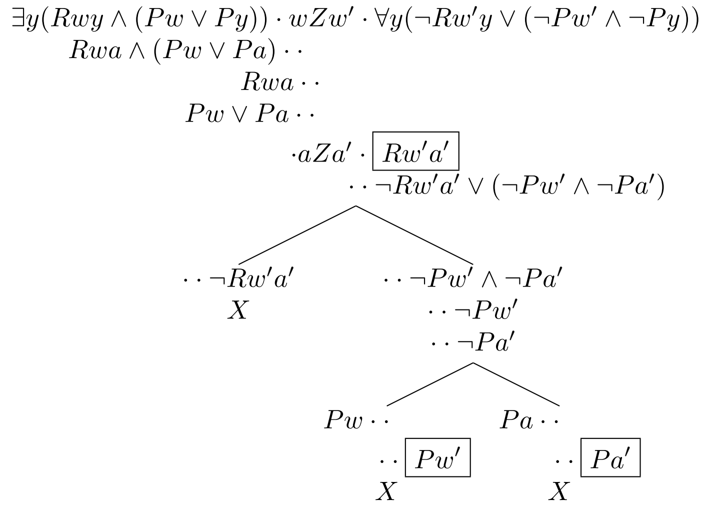
Example 1. Let $F(x) = \exists y \forall z (Rxy \wedge (\neg Ryz \vee Pz \vee Qz))$.


$\exists y \forall z (Rwy \wedge (\neg Ryz \vee Pz \vee Qz)) \cdot wZw' \cdot \forall y \exists z (\neg Rw'y \vee (Ryz \wedge \neg Pz \wedge \neg Qz))$
$\forall z (Rwa \wedge (\neg Raz \vee Pz \vee Qz)) \cdot \cdot$
$Rwa \wedge (\neg Raw \vee Pw \vee Qw) \cdot \cdot$
$Rwa \cdot \cdot$
$\neg Raw \vee Pw \vee Qw \cdot \cdot$
$\cdot aZa' \cdot \boxed{Rw'a'}$
$\cdot \cdot \exists z (\neg Rw'a' \vee (Ra'z \wedge \neg Pz \wedge \neg Qz))$
$\cdot \cdot \neg Rw'a' \vee (Ra'b' \wedge \neg Pb' \wedge \neg Qb')$
$\cdot \cdot \neg Rw'a'$
X
$\cdot \cdot Ra'b' \wedge \neg Pb' \wedge \neg Qb'$
$\cdot \cdot Ra'b'$
$\cdot \cdot \neg Pb'$
$\cdot \cdot \neg Qb'$
$\boxed{Rab} \cdot bZb' \cdot$
$Rwa \wedge (\neg Rab \vee Pb \vee Qb) \cdot \cdot$
$Rwa \cdot \cdot$
$\neg Rab \vee Pb \vee Qb \cdot \cdot$
$\neg Rab \cdot \cdot$
X
$Pb \cdot \cdot$
$\cdot \cdot \boxed{Pb'}$
X
$Qb \cdot \cdot$
$\cdot \cdot \boxed{Qb'}$
X

We have $F' = \exists y (Rxy \wedge \forall z (\neg Ryz \vee Pz \vee Qz))$, which is the standard translation of a modal formula $\Diamond \Box (p \vee q)$. Clearly, F' is equivalent to F.

Interim formula enables easier translation to a modal formula. In the previous example it is in fact a direct translation. This is not always the case.

Example 2. Let $F(x) = \exists y(Rxy \wedge (Px \vee Py))$.


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

We have $F' = \exists y(Rxy \wedge (Px \vee Py))$, which is actually equal to F. This is not exactly the standard translation of some modal formula. But, we have the following sequence of formulas clearly equivalent to F':

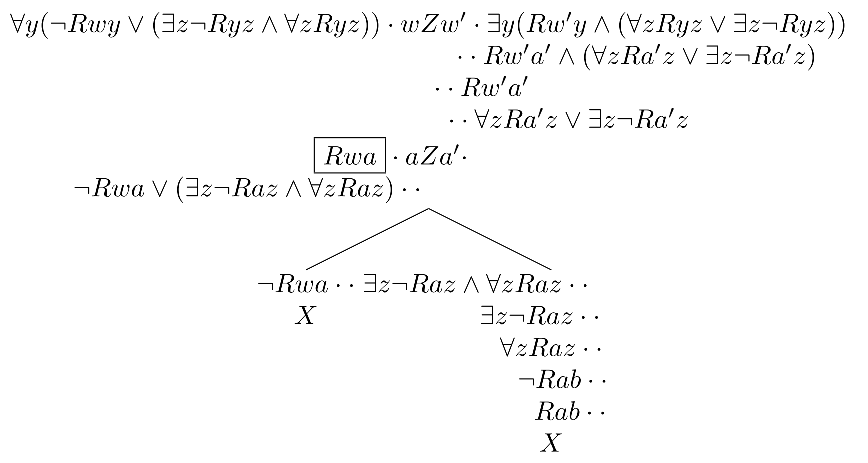
$\exists y((Rxy \wedge Px) \vee (Rxy \wedge Py))$
$\exists y(Rxy \wedge Px) \vee \exists y(Rxy \wedge Py)$
$(\exists y Rxy \wedge Px) \vee \exists y(Rxy \wedge Py)$

Now, the latter formula is the standard translation of $(\Diamond\top \wedge p) \vee \Diamond p$. The key to obtain this was to remove Px from the scope of $\exists y$.

As in the previous example, we can allways use basic equivalences of first-order formulas (cf. [6], p. 117) to obtain the standard translation of some modal formula from an interim formula.

Consider another example, which illustrates how interim formula can be significantly simpler then the initial formula, which was not the case in previous examples.

Example 3. Let $F(x) = \forall y(\neg Rxy \vee (\exists z \neg Ryz \wedge \forall z Ryz))$.


$\forall y(\neg Rwy \vee (\exists z \neg Ryz \wedge \forall z Ryz)) \cdot wZw' \cdot \exists y(Rw'y \wedge (\forall z Ryz \vee \exists z \neg Ryz))$
$\cdot \cdot Rw'a' \wedge (\forall z Ra'z \vee \exists z \neg Ra'z)$
$\cdot \cdot Rw'a'$
$\cdot \cdot \forall z Ra'z \vee \exists z \neg Ra'z$
$\boxed{Rwa} \cdot aZa' \cdot$
$\neg Rwa \vee (\exists z \neg Raz \wedge \forall z Raz) \cdot \cdot$
$\neg Rwa \cdot \cdot$
X
$\exists z \neg Raz \wedge \forall z Raz \cdot \cdot$
$\exists z \neg Raz \cdot \cdot$
$\forall z Raz \cdot \cdot$
$\neg Rab \cdot \cdot$
$Rab \cdot \cdot$
X

We have $F' = \forall y(\neg Rxy \vee \bot) = ST_x(\Box \bot)$.

Theorem 2. *Assume we have a closed BI-tableau for $F(x)$. Then the interim formula F' is equivalent to F. Furthermore, F' can be effectively rewriten to an equivalent formula that is the standard translation of some modal formula.*

Proof. Consider any node of the given BI-tableau. Let G' be the subformula of F' obtained by the procedure starting from that node. In what follows, we will treat two conclusions of $\wedge$-rule as one node (so, this is the only case in which we have more than one formula on left or right side). We claim that G' is equivalent to one of the following:

(1) $Q_1 x_{i+1} Q_2 x_{i+2} \dots G(x/w, x_1/a_1, x_2/a_2, \dots)$, where G is a formula or the conjunction of all formulas which occur on the left side or boxed on the right side of some nodes, with at least one of these nodes being between the previous branching and the current node, including the latter
(2) $Q_1 x_{i+1} Q_2 x_{i+2} \dots \neg G(x/w', x_1/a'_1, x_2/a'_2, \dots)$, with an analogous condition as in (1), just with left and right side switched,

where each of $Q_1, Q_2, \dots$ is $\forall$ or $\exists$ and $x_{i+1}, x_{i+2}, \dots$ are variables corresponding to constant symbols $a_{i+1}, a_{i+1}, \dots$ or $a'_{i+1}, a'_{i+1}, \dots$ occuring in G such that $a_{i+1} Z a'_{i+1}$, $a_{i+2} Z a'_{i+2}, \dots$ do not occur in the tableau prior to the current node.

Note that, applied to the root, this claim becomes the desired claim that F' is equivalent to F.

We prove the claim by induction on the distance of the current node to the farthest leaf.

In the base case, the current node is a leaf, so G' may be of the form Rx_ix_j, $\neg Rx_ix_j$, Px_i, $\neg Px_i$, $\top$ or $\bot$.

- $G' = Rx_ix_j$. This case occurs only if we have a contradiction between $\boxed{Ra'_ia'_j}$ and $\neg Ra'_ia'_j$ on the right side. So, $G = Ra'_ia'_j$ occurs boxed on the right side, while $G(x_i/a'_i, x_j/a'_j) = Rx_ix_j$ is equivalent (in fact, equal) to G'.
- $G' = \neg Rx_ix_j$ results from the occurrence of $\boxed{Ra_ia_j}$ and $G = \neg Ra_ia_j$ on the left side, so $G(x_i/a_i, x_j/a_j)$ is equivalent to (in fact, equals) $\neg Rx_ix_j = G'$.
- $G' = Px_i$ implies $\boxed{Pa'_i}$ and $\neg Pa'_i$ occur on the right side, so for $G = Pa'_i$ we have that G occurs boxed on the right side and $G(x_i/a'_i) = Px_i = G'$.
- $G' = \neg Px_i$ implies $\boxed{Pa_i}$ and $G = \neg Pa_i$ occur on the left side, so we have $G(x_i/a_i) = \neg Px_i = G'$.
- $G' = \top$ implies that some G and $\neg G$ occur on the right side on the path. Clearly, $\neg(G \wedge \neg G) \equiv \neg G \vee G$ is equivalent to G'. This remains to be the case if we prefix $\neg G \vee G$ with quantifiers as stated in (1) and (2).
- $G' = \bot$ implies that some G and $\neg G$ occur on the left side. Clearly, $G \wedge \neg G$ is equivalent to $G' = \bot$. Again, this holds regardless of any quantifiers prefixed to $G \wedge \neg G$.

Note that in all of the base cases, one of the nodes containing the involved formulas indeed occurs under the previous branching, since the branch is immediately closed after a contradiction occurs.

Inductive step has several cases, depending on the next rule applied in the tableau, starting from the current node.

- If the next rule applied in the tableau is $\forall$ or $\exists$-rule, which both have no immediate effect on F', the subformula of F' obtained starting after the application of this rule is again G'. By induction hypothesis, (1) or (2) applies on G', with G (or one of its conjuncts) occuring either prior to the current node, in which case the claim is proved, or exactly at the node succeding the application of $\forall$-rule. In the latter case, the claim clearly also holds, since (this conjunct of) G should be prefixed by Q_jx_j, where x_j is the variable involved in this application of $\forall$ or $\exists$-rule, and this prefixed formula occurs at the current node.
- As in the previous case, an application of $\wedge$-rule also has no immediate effect on F', i.e. the subformula of F' obtained starting after the application of $\wedge$-rule is still G'. By induction hypothesis, G' is implied by a formula of the form described in (1) or (2), where G or one of its conjuncts occur either above the current node, in which case the claim holds, or immediately after the application of $\wedge$-rule. But, since we treat conclusions of $\wedge$-rule as one node, taking the conjunction of formulas from that node is the same as taking the formula on which $\wedge$-rule was applied.
- In the case of the left $\vee$-rule, the current node is of the form $G_1 \vee G_2 \cdots$, while $G' = G'_1 \vee G'_2$, where G'_1 and G'_2 are subformulas obtained starting from the roots of two branches. By induction hypothesis, (1) applies to G'_1 and to G'_2, where G or one of its conjuncts is G_1 and G_2, respectively, since these are the only nodes after the previous branching. So clearly, $G_1 \vee G_2$ will work as G (or instead of a conjunct of G) for G'.

- In the case of the right $\vee$-rule, the current node is of the form $\cdot \cdot G_1 \vee G_2$ and $G' = G'_1 \wedge G'_2$, where G'_1 and G'_2 are obtained starting from the roots of branches. By induction hypothesis, (2) applies to G'_1 and to G'_2, where G or one of its conjuncts is G_1 and G_2, respectively, as in the previous case. It is easy to see that $G_1 \vee G_2$ will work as G (or instead of a conjunct of G) for G'. For example, if G'_1 is equivalent to $\neg G_1$ and G'_2 is equivalent to $\neg G_2$, then G' is equivalent to $\neg(G_1 \vee G_2)$. It is easy to see that this still works with additional conjuncts or quantifiers which may occur as described in (1) and (2).
- (at)-rule does not change the subformula obtained thereafter, so it is G'. Also, the new node contains only one formula which is a boxed version of a formula which already occured above on the other side, so induction hypothesis implies the claim.
- (forth)-rule has the same property as (at)-rule concerning the content of the new node, but the subformula obtained after its application is G'_1, where $G' = \exists x_k G'_1$. We apply the induction hypothesis to G_1. Then the same G from (1) and (2) which corresponds to G'_1 can be used for G', prefixed by the additional quantifier $\exists x_k$. The case of (back)-rule is proved analogously.

It remains to show that a modal correspondent is effectively obtainable from F'. Clearly, if F' is already the standard translation of some modal formula (or if F is, in which case we do not need to use BI-tableau in the first place), it is easy to reverse standard translation to obtain this modal formula. If not, it can be proved inductively that using basic equivalences of first-order logic we can effectively obtain (by anti-prenexing or miniscoping – see e.g. [4]) an equivalent formula that is the standard translation of some modal formula. The only problem is that F'_1 that appears in case of an application of (forth) or (back)-rule should contain only x_j and its successors to be translatable directly to modal formula. If it, however, contains some other variables, atomic subformulas in which these occur can always be removed from the scope of the quantifier, since it binds only x_j. We omit further details due to limited space.

4 Further work and acknowledgments

Further work should include all the technical details omitted in this report. Furthermore, we aim to implement the procedure in collaboration with associates participating in the project Formal Reasoning and Semantics (FORMALS), including Luka Mikec, to whom I am grateful for useful feedback during the preparation of this report.

References

1. J. van Benthem, *Exploring Logical Dynamics*, Studies in Logic, Language and Information, CSLI Publications & FoLLI, Stanford, 1996.
2. P. Blackburn, M. de Rijke, Y. Venema, *Modal Logic*, Cambridge University Press, 2001.

3. G. Boolos, Trees and Finite Satisfiability: Proof of a Conjecture of Burgess, *Notre Dame Journal of Formal Logic*, 25 (1984) 193-197.
4. J. Harrison, *Handbook of Practical Logic and Automated Reasoning*, Cambridge University Press, 2009.
5. T. Perkov, Tableau-based bisimulation invariance testing, *Reports on Mathematical Logic* 48 (2013) 101–115.
6. R. M. Smullyan, *First-Order Logic*, Springer-Verlag, 1968.