

Tableaux for public announcement logic

Philippe Balbiani

Institut de Recherche en Informatique de Toulouse – CNRS

Hans van Ditmarsch

Institut de Recherche en Informatique de Toulouse – CNRS

and

University of Otago

Andreas Herzig

Institut de Recherche en Informatique de Toulouse – CNRS

Tiago de Lima*

Eindhoven University of Technology

September 5, 2008

Abstract

Public announcement logic extends multi-agent epistemic logic with dynamic operators to model the informational consequences of announcements to the entire group of agents. In this article we propose a labelled tableau calculus for this logic, and show that it decides satisfiability of formulas in deterministic polynomial space. Since this problem is known to be PSPACE-complete, it follows that our proof method is optimal.

keywords: analytic tableaux; dynamic epistemic logics; public announcement logic; knowledge representation and reasoning; multi-agent systems.

1 Introduction

Public announcement logic (PAL) was originally proposed by Plaza (1989). This is one of a family of logics aiming at modelling dynamics of knowledge and belief in multi-agent settings. These formalisms, called dynamic epistemic logics (DELs), are used to model a number of epistemic scenarios and puzzles (see (Baltag et al., 1998), (Gerbrandy, 1999), (Kooi, 2007), and (van Benthem et al., 2006) for some examples). PAL is the simplest of them. It extends epistemic logic (EL) with dynamic operators $[\cdot]$ having formulas as arguments. The formula $[\varphi]\psi$ stands for ‘ ψ is true after the public announcement of φ ’. Being the simplest form of agent communication, public announcements are present in all DELs.

*Corresponding author: Tiago de Lima, Eindhoven University of Technology, PO Box 513, 5600 MB EINDHOVEN, Netherlands. Phone: +31 402472114. <http://home.tm.tue.nl/tlima>, <mailto:t.d.lima@tue.nl>.

Traditionally, proof systems for DELs are obtained by means of *reduction axioms*. In the particular case of PAL, they permit rewriting of each formula into an equivalent formula in EL. A proof system for PAL can therefore be obtained by applying in sequence reduction and then any proof system for EL — for instance as done in (Kooi, 2007). Nevertheless the translated formula may be exponentially larger than the original one. That is, PAL is strictly more succinct. This is the reason why PAL is considered to be more convenient than EL for reasoning about knowledge (van Benthem et al., 2006). Curiously however, satisfiability checking for PAL is, just as for EL, PSPACE-complete (Lutz, 2006). In this article we investigate a proof method that is an alternative to Lutz’s and that is based on semantic tableaux.

In Section 2 we present the syntax and semantics of public announcement logic. In Section 3 we present our tableau calculus for PAL and show soundness and completeness. In Section 4 we show that the calculus can be implemented to run in nondeterministic polynomial space, and therefore is an optimal method for satisfiability checking in PAL. In Section 5 we present tableau calculi for some variants of PAL. In Section 6 we discuss related works and draw conclusions.

2 Public announcement logic

Assume a finite set of agents A and a countably infinite set of atoms P .

DEFINITION 1 (THE LANGUAGE \mathcal{L}_{PAL})

The language \mathcal{L}_{PAL} of public announcement logic is inductively defined by the following BNF:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_a\varphi \mid [\varphi]\varphi$$

where a ranges over A , and p ranges over P .

A formula of the form $K_a\varphi$ is read ‘the agent a knows that φ ’, and a formula of the form $[\varphi]\psi$ is read ‘after the announcement of φ , ψ is true’. The formulas \top and \perp , and the connectives \vee , \rightarrow and \leftrightarrow are defined by usual abbreviations. In addition, we abbreviate by \widehat{K} the dual of K , and by $\langle \cdot \rangle$ the dual of $[\cdot]$. A formula of the form $\widehat{K}_a\varphi$ should be read ‘the agent a considers it possible that φ ’, and a formula of the form $\langle \varphi \rangle \psi$ should be read ‘the announcement φ can be made and after that ψ is true’.

The formulas in \mathcal{L}_{PAL} are interpreted in Kripke structures called epistemic models.

DEFINITION 2 (EPISTEMIC MODEL)

An *epistemic model* is a triple $\langle W, R, V \rangle$ such that: W is a non-empty set of *possible worlds*; $R : A \rightarrow \wp(W \times W)$ associates a reflexive, transitive and symmetric accessibility relation $R(a)$ to each agent a in A ; and $V : P \rightarrow \wp(W)$ associates a *valuation* $V(p)$ to each atom p in P . If $M = \langle W, R, V \rangle$ is an epistemic model and $w \in W$, then the pair $\langle M, w \rangle$ is a *pointed epistemic model* (also called *epistemic state*).

For convenience we write $R(a)$ as R_a , and $V(p)$ as V_p . In addition, we denote by $R_a(w)$ the set $\{w' \mid \langle w, w' \rangle \in R_a\}$.

DEFINITION 3 (THE SATISFACTION RELATION)

Let $\langle M, w \rangle$ be an epistemic state. The satisfaction relation \models is defined inductively by:

$$\begin{aligned}
\langle M, w \rangle \models p & \quad \text{iff} \quad w \in V_p \\
\langle M, w \rangle \models \neg\varphi & \quad \text{iff} \quad \text{not } \langle M, w \rangle \models \varphi \\
\langle M, w \rangle \models \varphi \wedge \psi & \quad \text{iff} \quad \langle M, w \rangle \models \varphi \text{ and } \langle M, w \rangle \models \psi \\
\langle M, w \rangle \models K_a\varphi & \quad \text{iff} \quad \text{for all } w' \in R_a(w), \langle M, w' \rangle \models \varphi \\
\langle M, w \rangle \models [\varphi]\psi & \quad \text{iff} \quad \langle M, w \rangle \models \varphi \text{ implies } \langle M^\varphi, w \rangle \models \psi
\end{aligned}$$

where the epistemic model M^φ in the last clause above results from the update of M by the public announcement of φ , defined by the triple $\langle W^\varphi, R^\varphi, V^\varphi \rangle$ such that:

$$\begin{aligned}
W^\varphi & = \{w' \in W \mid \langle M, w' \rangle \models \varphi\} \\
R_a^\varphi & = R_a \cap (W^\varphi \times W^\varphi) \\
V_p^\varphi & = V_p \cap W^\varphi
\end{aligned}$$

That is, the epistemic model M^φ is the epistemic model M restricted to the worlds where φ is true. Then $\langle M, w \rangle \models \langle \varphi \rangle \psi$ if and only if $\langle M, w \rangle \models \varphi$ and $\langle M^\varphi, w \rangle \models \psi$. The dynamic modal operator $[\cdot]$ is thus interpreted as an epistemic state transformer. Announcements are assumed to be truthful, and are commonly perceived by all agents in A . For simplicity, given a list of public announcements (ψ_1, \dots, ψ_k) , we abbreviate by $M^{(\psi_1, \dots, \psi_k)}$ the epistemic model $(\dots (M^{\psi_1}) \dots)^{\psi_k}$. Therefore, with ϵ denoting the empty list, we have $M^\epsilon = M$.

DEFINITION 4 (VALIDITY AND SATISFIABILITY)

Let $\varphi \in \mathcal{L}_{\text{PAL}}$. The formula φ is *valid in an epistemic model* M (notation: $M \models \varphi$), if and only if for all $w \in W$, $\langle M, w \rangle \models \varphi$; the formula φ is *valid* (notation: $\models \varphi$) if and only if for all epistemic models M , $M \models \varphi$; and the formula φ is *satisfiable* if and only if $\not\models \neg\varphi$ (i.e., $\neg\varphi$ is not valid).

EXAMPLE 5

Consider the sentence: ‘ p is true and you (agent a) do not know it’. Such a so-called Moore sentence, when true, has the interesting property of becoming false just after being publicly announced. To see it, note that by its announcement all the agents, in particular the agent a , learn that p . It immediately follows that, after the announcement, the agent a does know p . This interesting phenomenon can be modelled in public announcement logic by the valid formula $[\varphi]\neg\varphi$, where $\varphi = p \wedge \neg K_a p$. For the proof of its validity, let $\langle M, w \rangle$ be an arbitrary epistemic state. If $\langle M, w \rangle \models \varphi$, then $\langle M, w \rangle \models p$, which implies that $\langle M^\varphi, w \rangle \models K_a p$, and therefore $\langle M^\varphi, w \rangle \models \neg\varphi$.

Since the class of models of PAL is the same as of that the logic $S5_n$, the following axiom schemata are also valid in PAL:

- K. $K_a(\varphi \rightarrow \psi) \rightarrow (K_a\varphi \rightarrow K_a\psi)$
- T. $K_a\varphi \rightarrow \varphi$
- 4. $K_a\varphi \rightarrow K_a K_a\varphi$
- 5. $\neg K_a\varphi \rightarrow K_a \neg K_a\varphi$

3 Tableau method

We present in this section a proof method for public announcement logic that uses analytic tableaux. As a typical tableau method, given a formula φ , it systematically tries to construct a model for it. When it fails, φ is inconsistent and thus its negation is valid.

In our notation each formula in the tableau is prefixed by a natural number that stands for a possible world in the model under construction, similar to the notation used by Fitting (1983, Chapter 8). In addition, each formula is prefixed by a list of announcements representing successive updates.

DEFINITION 6 (LABELLED FORMULA)

A *labelled formula* is a triple of the form $\langle \mu, n, \varphi \rangle$ such that: μ is a (possibly empty) list of formulas of \mathcal{L}_{PAL} ; $n \in \mathbb{N}$; and $\varphi \in \mathcal{L}_{\text{PAL}}$.

DEFINITION 7 (BRANCH)

A *branch* is a pair of the form $\langle L, S \rangle$ such that: L is a set of labelled formulas; and $S \subseteq (A \times \mathbb{N} \times \mathbb{N})$.

The relation S present in the branch is called *skeleton*. It represents (a part of) the relation R in the model.

DEFINITION 8 (TABLEAU)

Let $\varphi \in \mathcal{L}_{\text{PAL}}$. A *tableau for φ* is a set of branches T inductively defined as follows:

- $T = \{\langle \langle \epsilon, 0, \varphi \rangle, \emptyset \rangle\}$, where ϵ denotes the empty list. This is called the *initial tableau for φ* .
- $T = (T' \setminus \{b\}) \cup B$, where T' is a tableau for φ that contains the branch $b = \langle L, S \rangle$, and B is a finite set of branches generated by one of the *tableau rules* defined below, where \bullet denotes concatenation of lists:¹

Rcut1. if $\langle \mu, n, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L$ or $\langle \mu, n, [\varphi_1]\varphi_2 \rangle \in L$, then

$$B = \{\langle L \cup \{\langle \mu, n, \neg\varphi_1 \rangle, \langle \mu, n, \neg\varphi_2 \rangle\}, S \rangle, \langle L \cup \{\langle \mu, n, \neg\varphi_1 \rangle, \langle \mu, n, \varphi_2 \rangle\}, S \rangle, \langle L \cup \{\langle \mu, n, \varphi_1 \rangle, \langle \mu, n, \neg\varphi_2 \rangle\}, S \rangle, \langle L \cup \{\langle \mu, n, \varphi_1 \rangle, \langle \mu, n, \varphi_2 \rangle\}, S \rangle\}.$$

Rcut2. if $\langle \mu, n, \neg[\varphi_1]\varphi_2 \rangle \in L$ or $\langle \mu, n, K_a\varphi_2 \rangle \in L$ or $\langle \mu, n, \neg K_a\varphi_2 \rangle \in L$, then

$$B = \{\langle L \cup \{\langle \mu, n, \neg\varphi_2 \rangle\}, S \rangle, \langle L \cup \{\langle \mu, n, \varphi_2 \rangle\}, S \rangle\}.$$

RSB. if $\langle \mu \bullet (\psi), n, \ell \rangle \in L$ for some literal ℓ , then

$$B = \{\langle L \cup \{\langle \mu, n, \ell \rangle\}, S \rangle\}.$$

R \neg . if $\langle \mu, n, \neg\neg\varphi \rangle \in L$, then

$$B = \{\langle L \cup \{\langle \mu, n, \varphi \rangle\}, S \rangle\}.$$

R \wedge . if $\langle \mu, n, \varphi_1 \wedge \varphi_2 \rangle \in L$, then

$$B = \{\langle L \cup \{\langle \mu, n, \varphi_1 \rangle, \langle \mu, n, \varphi_2 \rangle\}, S \rangle\}.$$

R \vee . if $\langle \mu, n, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L$, then

$$B = \{\langle L \cup \{\langle \mu, n, \neg\varphi_1 \rangle\}, S \rangle, \langle L \cup \{\langle \mu, n, \neg\varphi_2 \rangle\}, S \rangle\}.$$

¹For those used to the more standard “nominator/denominator” form of tableau rules, it may be helpful to look at an alternative presentation of these rules given in Appendix A.

RT. if $\langle \mu, n, K_a \varphi \rangle \in L$, then
 $B = \{ \langle L \cup \{ \langle \mu, n, \varphi \rangle \}, S \rangle \}$.

RK. if $\langle (\psi_1, \dots, \psi_k), n, K_a \varphi \rangle \in L$ and $\langle a, n, n' \rangle \in S$, then
 $B = \{ L \cup \{ \langle \epsilon, n', [\psi_1] \dots [\psi_k] \varphi \rangle \}, S \}$.

R4. if $\langle (\psi_1, \dots, \psi_k), n, K_a \varphi \rangle \in L$ and $\langle a, n, n' \rangle \in S$, then
 $B = \{ L \cup \{ \langle \epsilon, n', [\psi_1] \dots [\psi_k] K_a \varphi \rangle \}, S \}$.

R5. if $\langle (\psi_1, \dots, \psi_k), n, \neg K_a \varphi \rangle \in L$ and $\langle a, n, n' \rangle \in S$, then
 $B = \{ \langle L \cup \{ \langle \epsilon, n', [\psi_1] \dots [\psi_k] \neg K_a \varphi \rangle \}, S \}$.

\widehat{RK} . if $\langle (\psi_1, \dots, \psi_k), n, \neg K_a \varphi \rangle \in L$, then
 $B = \{ \langle L \cup \{ \langle \epsilon, n', \neg [\psi_1] \dots [\psi_k] \varphi \rangle \}, S \cup \{ \langle a, n, n' \rangle \} \}$ for some $n' \in \mathbb{N}$ not occurring in S .

$R[\cdot]$. if $\langle \mu, n, [\varphi_1] \varphi_2 \rangle \in L$, then
 $B = \{ \langle L \cup \{ \langle \mu, n, \neg \varphi_1 \rangle \}, S \rangle, \langle L \cup \{ \langle \mu, n, \varphi_1 \rangle, \langle \mu \bullet (\varphi_1), n, \varphi_2 \rangle \}, S \rangle \}$.

$R\langle \cdot \rangle$. if $\langle \mu, n, \neg [\varphi_1] \varphi_2 \rangle \in L$, then
 $B = \{ \langle L \cup \{ \langle \mu, n, \varphi_1 \rangle, \langle \mu \bullet (\varphi_1), n, \neg \varphi_2 \rangle \}, S \rangle \}$.

Rules Rcut1 and Rcut2 are bounded cuts (also called semi-analytic cuts). The *step-back* rule RSB reflects the fact that the valuations of the atoms do not change through announcements. Rules $R\neg$, $R\wedge$ and $R\vee$ are exactly as for propositional logic. Rule $R[\cdot]$ reflects the semantics of the public announcement operator: by definition an epistemic state satisfies $[\varphi_1] \varphi_2$ if and only if it satisfies $\neg \varphi_1$, or it satisfies φ_1 and its restriction to φ_1 satisfies φ_2 . Rule $R\langle \cdot \rangle$ is the dual of rule $R[\cdot]$. Rule RT corresponds to reflexivity, which by its turn corresponds to the axiom scheme T.

The other four rules involving the operator K are different from their counterparts commonly used in tableau calculi for epistemic logics. We start by giving the intuition behind rule \widehat{RK} : let $\langle M^\mu, w \rangle$ be the epistemic state corresponding to the pair $\langle \mu, n \rangle$. If $\langle M^\mu, w \rangle$ satisfies the formula $\neg K_a \varphi$, then there must exist a world w' (corresponding to n' in the tableau) in $R_a(w)$ such that $\langle M^\mu, w' \rangle$ satisfies $\neg \varphi$. But M^μ is the epistemic model M updated by the sequence of announcements μ . This means that w' must in addition be present in the epistemic model before the announcements (i.e., in the epistemic model M), and moreover should “survive” all the updates in μ . In other words, if $\mu = (\psi_1, \dots, \psi_k)$, then $\langle M, w' \rangle$ should satisfy ψ_1 , and $\langle M^{\psi_1}, w' \rangle$ should satisfy ψ_2 , ..., and $\langle M^{(\psi_1, \dots, \psi_{k-1})}, w' \rangle$ should satisfy ψ_k . This is equivalent to state that $\langle M, w' \rangle$ should satisfy $\langle \psi_1 \rangle \dots \langle \psi_i \rangle \neg \varphi$. The latter formula is equivalent to $\neg [\psi_1] \dots [\psi_2] \varphi$. Now, for the intuition behind rule RK, suppose that the epistemic model $\langle M^\mu, w \rangle$ satisfies $K_a \varphi$, and that there exists $w' \in R_a(w)$. Then, if w' “survives” the sequence of updates in μ , $\langle M^\mu, w' \rangle$ should satisfy φ . That is, $\langle M, w' \rangle$ should satisfy the formula $[\psi_1] \dots [\psi_i] \varphi$. Rule R4 implements the same idea, as well as R5. The former corresponds to transitivity, while the latter (together with the bounded cuts) corresponds to symmetry.

DEFINITION 9 (CLOSED TABLEAU)

Let $b = \langle L, S \rangle$ be a branch. The set L is *closed* if and only if $\{ \langle \mu, n, \varphi \rangle, \langle \mu, n, \neg \varphi \rangle \} \subseteq L$ for some n, μ and φ . The branch b is closed if and only if L is closed. The

1.	$\langle \epsilon, 0, \neg[p \wedge \neg K_a p] \neg(p \wedge \neg K_a p) \rangle$	
2.	$\langle \epsilon, 0, p \wedge \neg K_a p \rangle$	(R $\langle \cdot \rangle$: 1)
3.	$\langle (p \wedge \neg K_a p), 0, \neg \neg(p \wedge \neg K_a p) \rangle$	(R $\langle \cdot \rangle$: 1)
4.	$\langle (p \wedge \neg K_a p), 0, p \wedge \neg K_a p \rangle$	(R \neg : 3)
5.	$\langle (p \wedge \neg K_a p), 0, p \rangle$	(R \wedge : 4)
6.	$\langle (p \wedge \neg K_a p), 0, \neg K_a p \rangle$	(R \wedge : 4)
7.	$\langle \epsilon, 1, \neg[p \wedge \neg K_a p] p \rangle$	$\langle a, 0, 1 \rangle \in S$ (R \widehat{K} : 6)
8.	$\langle \epsilon, 1, p \wedge \neg K_a p \rangle$	(R $\langle \cdot \rangle$: 7)
9.	$\langle (p \wedge \neg K_a p), 1, \neg p \rangle$	(R $\langle \cdot \rangle$: 7)
10.	$\langle \epsilon, 1, p \rangle$	(R \wedge : 8)
11.	$\langle \epsilon, 1, \neg K_a p \rangle$	(R \wedge : 8)
12.	$\langle \epsilon, 1, \neg p \rangle$	(RSB : 9)
	closed	(10, 12)

Figure 1: Closed tableau for the formula $[p \wedge \neg K_a p] \neg(p \wedge \neg K_a p)$.

branch b is *open* if and only if it is not closed. A tableau is closed if and only if all its branches are closed. A tableau is open if and only if it is not closed.

EXAMPLE 10

In Figure 1 the tableau method is used to show the validity of the formula $[p \wedge \neg K_a p] \neg(p \wedge \neg K_a p)$ of Example 5. The rightmost column shows which tableau rule is applied in each line.

We proceed by proving soundness. But first we need yet another definition.

DEFINITION 11 (SATISFIABLE BRANCH)

The branch $b = \langle L, S \rangle$ is *satisfiable* if and only if there exists an epistemic model $M = \langle W, R, V \rangle$ and a function $f : \mathbb{N} \rightarrow W$ such that $\langle f(n), f(n') \rangle \in R_a$ for all $\langle a, n, n' \rangle \in S$, and for all $\langle (\psi_1, \dots, \psi_k), n, \varphi \rangle \in L$: $\langle M, f(n) \rangle \models \psi_1$, $\langle M^{\psi_1}, f(n) \rangle \models \psi_2, \dots, \langle M^{(\psi_1, \dots, \psi_{k-1})}, f(n) \rangle \models \psi_k$ and $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models \varphi$.

Note that if a branch containing the labelled formula $\langle (\psi_1, \dots, \psi_k), n, \varphi \rangle$ is satisfiable, then $f(n) \in (W \cap W^{\psi_1} \cap \dots \cap W^{(\psi_1, \dots, \psi_k)})$.

THEOREM 12 (SOUNDNESS)

If there exists a closed tableau for $\neg\varphi_0$, then φ_0 is valid.

PROOF. We show that if φ_0 is satisfiable, then there is no closed tableau for φ_0 . It is enough to show that all tableau rules preserve satisfiability. That is, it is enough to show that: if the branch $b = \langle L, S \rangle$ is satisfiable, then the set of branches B generated by any tableau rule contains a satisfiable branch. To see it, suppose that b is satisfiable and closed. Then L contains two labelled formulas, one of the form $\langle \mu, n, \varphi \rangle$ and another of the form $\langle \mu, n, \neg\varphi \rangle$. Because b is satisfiable, there exists an epistemic model M and a function f such that $\langle M^\mu, f(n) \rangle \models \varphi$ and $\langle M^\mu, f(n) \rangle \models \neg\varphi$, which is a contradiction.

Now, suppose that the branch b is satisfiable. The rules Rcut1, Rcut2, R \neg , R \wedge and R \vee are analogous to the rules commonly used for propositional logic. The proof that they preserve satisfiability is straightforward.

We proceed by showing that rule RSB preserves satisfiability: first, suppose that $\ell = p$. $\langle M^{\mu \bullet (\psi)}, f(n) \rangle \models p$ (by hypothesis) iff $f(n) \in V_p^{\mu \bullet (\psi)}$. Then $f(n) \in V_p^\mu$ (by Definition 3) iff $\langle M^\mu, f(n) \rangle \models p$. The case for $\ell = \neg p$ is analogous.

RT: $\langle M^\mu, f(n) \rangle \models K_a \varphi$ (by hypothesis), then $\langle M^\mu, f(n) \rangle \models \varphi$ (by axiom scheme T).

RK: $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models K_a \varphi$ (by hypothesis) iff $f(n) \in R_a^{(\psi_1, \dots, \psi_k)}(f(n))$ implies $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models \varphi$ iff $f(n) \in R_a^{(\psi_1, \dots, \psi_{k-1})}(f(n))$ and $\langle M^{(\psi_1, \dots, \psi_{k-1})}, f(n) \rangle \models \psi_k$, implies $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models \varphi$ (by Definition 3) iff $f(n) \in R_a^{(\psi_1, \dots, \psi_{k-1})}(f(n))$ implies $\langle M^{(\psi_1, \dots, \psi_{k-1})}, f(n) \rangle \models [\psi_k] \varphi$ (by Definition 3). Applying the same argument k times, we have that: iff $f(n) \in R_a(f(n))$ implies $\langle M, f(n) \rangle \models [\psi_1] \dots [\psi_k] \varphi$.

R4: $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models K_a \varphi$ (by hypothesis), then $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models K_a K_a \varphi$ (by axiom scheme 4) iff $f(n) \in R_a(f(n))$ implies $\langle M, f(n) \rangle \models [\psi_1] \dots [\psi_k] K_a \varphi$ (by the same argument as used for rule RK).

R5: $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models \neg K_a \varphi$ (by hypothesis), then $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models K_a \neg K_a \varphi$ (by axiom scheme 5) iff $f(n) \in R_a(f(n))$ implies $\langle M, f(n) \rangle \models [\psi_1] \dots [\psi_k] \neg K_a \varphi$ (by the same argument as used for rule RK).

R \widehat{K} : $\langle M^{(\psi_1, \dots, \psi_k)}, f(n) \rangle \models \neg K_a \varphi$ (by hypothesis) then there exists $w' \in R_a^{(\psi_1, \dots, \psi_k)}(f(n))$ such that $\langle M^{(\psi_1, \dots, \psi_k)}, w' \rangle \models \neg \varphi$ iff $\langle M, w' \rangle \models \psi_1$ and $\langle M^{\psi_1}, w' \rangle \models \psi_2$ and ... and $\langle M^{(\psi_1, \dots, \psi_{k-1})}, w' \rangle \models \psi_k$ and $\langle M^{(\psi_1, \dots, \psi_k)}, w' \rangle \models \neg \varphi$ (by Definition 3 applied k times) iff $\langle M, w' \rangle \models \neg [\psi_1] \dots [\psi_k] \varphi$ (again, by Definition 3 applied k times). Now, consider the function $f' : \mathbb{N} \rightarrow W$ such that for all n occurring in L , $f'(n) = f(n)$, and $f'(n') = w'$. Then for all $\langle \mu'', n'', \varphi'' \rangle \in L$, $\langle M^{\mu''}, f'(n'') \rangle \models \varphi''$ (because n' does not occur in L) and $\langle M, f'(n') \rangle \models \neg [\psi_1] \dots [\psi_k] \varphi$.

R[·]: $\langle M^\mu, f(n) \rangle \models [\varphi_1] \varphi_2$ (by hypothesis) iff $\langle M^\mu, f(n) \rangle \models \neg \varphi_1$, or $\langle M^\mu, f(n) \rangle \models \varphi_1$ and $\langle M^{\mu \bullet (\varphi_1)}, f(n) \rangle \models \varphi_2$ (by Definition 3). Therefore, one of the branches in B is satisfiable.

R(·): $\langle M^\mu, f(n) \rangle \models \neg [\varphi_1] \varphi_2$ (by hypothesis) iff $\langle M^\mu, f(n) \rangle \models \varphi_1$ and $\langle M^{\mu \bullet (\varphi_1)}, f(n) \rangle \models \varphi_2$ (by Definition 3). ■

In the rest of the section we prove completeness. First though, we need another auxiliary definition.

DEFINITION 13 (SATURATED TABLEAU)

Let T be a tableau for φ . T is *saturated* if and only if T is saturated under all tableau rules, as defined below:

1. T is *saturated under rule Rcut1* if and only if for all $b = \langle L, S \rangle \in T$, if $\langle \mu, n, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L$ or $\langle \mu, n, [\varphi_1] \varphi_2 \rangle \in L$, then $\{\langle \mu, n, \neg \varphi_1 \rangle, \langle \mu, n, \neg \varphi_2 \rangle\} \subseteq L$ or $\{\langle \mu, n, \neg \varphi_1 \rangle, \langle \mu, n, \varphi_2 \rangle\} \subseteq L$ or $\{\langle \mu, n, \varphi_1 \rangle, \langle \mu, n, \neg \varphi_2 \rangle\} \subseteq L$ or $\{\langle \mu, n, \varphi_1 \rangle, \langle \mu, n, \varphi_2 \rangle\} \subseteq L$.
2. T is *saturated under rule Rcut2* if and only if for all $b = \langle L, S \rangle \in T$, if $\langle \mu, n, \neg[\varphi_1] \varphi_2 \rangle \in L$ or $\langle \mu, n, K_a \varphi_2 \rangle \in L$ or $\langle \mu, n, \neg K_a \varphi_2 \rangle \in L$, then $\langle \mu, n, \neg \varphi_2 \rangle \in L$ or $\langle \mu, n, \varphi_2 \rangle \in L$.
3. T is *saturated under rule R \neg* if and only if for all $b = \langle L, S \rangle \in T$, if $\langle \mu, n, \neg \neg \varphi \rangle \in L$, then $\langle \mu, n, \varphi \rangle \in L$.

4. T is saturated under rule $R\wedge$ if and only if for all $b = \langle L, S \rangle \in T$, if $\langle \mu, n, \varphi_1 \wedge \varphi_2 \rangle \in L$, then $\{\langle \mu, n, \varphi_1 \rangle, \langle \mu, n, \varphi_2 \rangle\} \subseteq L$.
5. T is saturated under rule $R\vee$ if and only if for all $b = \langle L, S \rangle \in T$, if $\langle \mu, n, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L$, then $\langle \mu, n, \neg\varphi_1 \rangle \in L$ or $\langle \mu, n, \neg\varphi_2 \rangle \in L$.
6. T is saturated under rule RK if and only if for all $b = \langle L, S \rangle \in T$, if $\langle (\psi_1, \dots, \psi_k), n, K_a\varphi \rangle \in L$ and $\langle a, n, n' \rangle \in S$, then $\langle \epsilon, n, [\psi_1] \dots [\psi_k]\varphi \rangle \in L$.
7. T is saturated under rule RT if and only if for all $b = \langle L, S \rangle \in T$, if $\langle (\psi_1, \dots, \psi_k), n, K_a\varphi \rangle \in L$, then $\langle \epsilon, n, [\psi_1] \dots [\psi_k]\varphi \rangle \in L$.
8. T is saturated under rule $R4$ if and only if for all $b = \langle L, S \rangle \in T$, if $\langle (\psi_1, \dots, \psi_k), n, K_a\varphi \rangle \in L$ and $\langle a, n, n' \rangle \in S$, then $\langle \epsilon, n', [\psi_1] \dots [\psi_k]K_a\varphi \rangle \in L$.
9. T is saturated under rule $R5$ if and only if for all $b = \langle L, S \rangle \in T$, if $\langle (\psi_1, \dots, \psi_k), n, \neg K_a\varphi \rangle \in L$ and $\langle a, n, n' \rangle \in S$, then $\langle \epsilon, n', [\psi_1] \dots [\psi_k]\neg K_a\varphi \rangle \in L$.
10. T is saturated under rule $R\widehat{K}$ if and only if for all $b = \langle L, S \rangle \in T$, if $\langle (\psi_1, \dots, \psi_k), n, \neg K_a\varphi \rangle \in L$, then $\langle \epsilon, n', \neg[\psi_1] \dots [\psi_k]\varphi \rangle \in L$ and $\langle a, n, n' \rangle \in S$.
11. T is saturated under rule $R[\cdot]$ if and only if for all $b = \langle L, S \rangle \in T$, if $\langle \mu, n, [\varphi_1]\varphi_2 \rangle \in L$, then $\{\langle \mu, n, \neg\varphi_1 \rangle\} \subseteq L$ or $\{\langle \mu, n, \varphi_1 \rangle, \langle \mu \bullet (\varphi_1), n, \varphi_2 \rangle\} \subseteq L$.
12. T is saturated under rule $R\langle \cdot \rangle$ if and only if for all $b = \langle L, S \rangle \in T$, if $\langle \mu, n, \neg[\varphi_1]\varphi_2 \rangle \in L$, then $\{\langle \mu, n, \varphi_1 \rangle, \langle \mu \bullet (\varphi_1), n, \varphi_2 \rangle\} \subseteq L$.

THEOREM 14 (COMPLETENESS)

If φ_0 is valid, then there exists a closed tableau for $\neg\varphi_0$.

PROOF. The following two lemmas will be used in this proof.

LEMMA 15

Let $\langle L, S \rangle$ be a branch of a tableau T for φ . If $\langle (\psi_1, \dots, \psi_k), n, \varphi \rangle \in L$, then $\langle (\psi_1, \dots, \psi_{i-1}), n, \psi_i \rangle \in L$ for all $1 \leq i \leq k$, where $\langle \psi_1, \dots, \psi_{i-1} \rangle = \epsilon$ if $i = 1$.

PROOF. Let T_0, T_1, \dots, T_m be a sequence of tableaux for φ such that T_0 is the initial tableau for φ , $T_m = T$, and each T_j is obtained from T_{j-1} by the application of one of the tableau rules. We show this lemma by induction on m .

The base case, i.e. $m = 0$, follows immediately from the definition of initial tableau for φ . For the induction step consider the branch b in the tableau T_{m+1} . If it contains a labelled formula of the form $\langle (\psi_1, \dots, \psi_k), n, \varphi \rangle$, then either it is in a branch of T_m , and in this case the induction hypothesis applies, or it is added by the application of one of the tableau rules in T_m . Note that it cannot be added by the application of RK , RT , $R4$, $R5$ or $R\widehat{K}$. If it is added by the application of $R\text{cut}1$, $R\text{cut}2$, $R\neg$, $R\wedge$ or $R\vee$, then T_m contains a formula of the form $\langle (\psi_1, \dots, \psi_k), n', \varphi' \rangle$ and the induction hypothesis applies. Else if it is added by the application of $R[\cdot]$ or $R\langle \cdot \rangle$, then we have one of the following two cases: (i) the labelled formula $\langle (\psi_1, \dots, \psi_k), n, \varphi \rangle$ is present in T_m , or

(ii) the labelled formula $\langle\langle\psi, \dots, \psi_{k-1}, n, \varphi\rangle\rangle$ is present in T_m and the formula $\langle\langle\psi_1, \dots, \psi_{k-1}, n, \psi_k\rangle\rangle$ is added to T_{m+1} . In both cases, the induction hypothesis applies. ■

LEMMA 16

Let $b = \langle L, S \rangle$ be a branch of an open saturated tableau for φ_0 . If $\langle\mu, n, \varphi\rangle \in L$, then for every sub-formula φ_1 of φ , either $\langle\mu, n, \varphi_1\rangle \in L$ or $\langle\mu, n, \neg\varphi_1\rangle \in L$.

PROOF. By induction on the structure of φ . Let T be an open saturated tableau for φ_0 . The two base cases, that is, $\varphi = p$ and $\varphi = \neg p$ for some $p \in P$, follow immediately. In the induction step, for the cases $\varphi = \neg\neg\varphi_1$ and $\varphi = \varphi_1 \wedge \varphi_2$, we use that T is saturated respectively under the rules $R\neg$ and $R\wedge$. For the cases $\varphi = \neg(\varphi_1 \wedge \varphi_2)$ and $\varphi = [\varphi_1]\varphi_2$, we use that T is saturated under rule $Rcut1$. For the cases $\varphi = K_a\varphi_1$ and $\varphi = \neg K_a\varphi_1$, we use that T is saturated under rule $Rcut2$. And for the case $\varphi = \neg[\varphi_1]\varphi_2$, we use that T is saturated under the rules $Rcut1$ and $R\langle\cdot\rangle$. ■

Now we are ready to prove completeness. We do so by showing that if there exists an open saturated tableau for $\varphi_0 \in \mathcal{L}_{PAL}$, then φ_0 is satisfiable. Suppose that T is an open saturated tableau for φ_0 . Then, it contains at least one open branch $b = \langle L, S \rangle$. We use this branch to construct an epistemic model $M = \langle W, R, V \rangle$ as follows: $W = \{n \in \mathbb{N} \mid \langle\mu, n, \varphi\rangle \in L \text{ for some } \mu, \varphi\}$; each R_a is the reflexive, transitive and symmetric closure of $\{\langle n, n' \rangle \mid \langle a, n, n' \rangle \in S\}$; and each $V_p = \{n \mid \langle \epsilon, n, p \rangle \in L\}$. Clearly, W is a non-empty set, each R_a is an equivalence relation, and V assigns a subset of W to each proposition letter p occurring in L .

We now show that for all labelled formulas $\lambda = \langle\langle\psi_1, \dots, \psi_k\rangle, n, \varphi\rangle \in L$:

$$\mathcal{P}(\lambda) = \begin{cases} \langle M, n \rangle \models \psi_1 & \text{and} \\ \langle M^{\psi_1}, n \rangle \models \psi_2 & \text{and} \\ \vdots & \\ \langle M^{(\psi_1, \dots, \psi_{k-1})}, n \rangle \models \psi_k & \text{and} \\ \langle M^{(\psi_1, \dots, \psi_k)}, n \rangle \models \varphi & \end{cases}$$

holds by induction on the size of the sequence of announcements k .

The base case is $k = 0$. That is, λ is of the form $\langle\epsilon, n, \varphi\rangle$. We show $\mathcal{P}(\lambda)$ also by induction, but now on the structure of φ .

There are two base cases. First suppose that $\varphi = p$ for some $p \in P$: $\lambda \in L$ iff $n \in V_p$ iff $\langle M, n \rangle \models p$. The case $\varphi = \neg p$ is analogous.

For the first case of the induction step suppose that $\varphi = \neg\neg\varphi_1$: iff $\langle\epsilon, n, \varphi_1\rangle \in L$ (because T is saturated under rule $R\neg$). Then $\langle M, n \rangle \models \varphi_1$ (by induction hypothesis), iff $\langle M, n \rangle \models \neg\neg\varphi_1$.

The cases $\varphi = \varphi_1 \wedge \varphi_2$ and $\varphi = \neg(\varphi_1 \wedge \varphi_2)$ are straightforward.

We proceed with the case $\varphi = K_a\varphi_1$. We have to show that for all $n' \in R_a(n)$, $\langle M, n' \rangle \models \varphi_1$. This is true if for all $n' \in R_a(n)$, $\langle\epsilon, n', \varphi_1\rangle \in L$ (by induction hypothesis). Since R_a is the reflexive, transitive and symmetric closure of $\{\langle n, n' \rangle \mid \langle a, n, n' \rangle \in S\}$, it amounts to show that: (i) $\langle\epsilon, n, \varphi_1\rangle \in L$; (ii) for all a -descendants n' of n in S , $\langle\epsilon, n', \varphi_1\rangle \in L$. That is, it holds for all n_i in the sequence $n = n_0, n_1, \dots, n_m$ such that $\langle a, n_i, n_{i+1} \rangle \in S$, where $0 \leq i \leq m$, and where there is no n'' such that $\langle a, n_m, n'' \rangle \in S$; (iii) for all

a -antecedents n' of n in S , $\langle \epsilon, n', \varphi_1 \rangle \in L$. That is, it holds for all n_i in the sequence $n = n_m, \dots, n_1, n_0$ such that $\langle a, n_i, n_{i+1} \rangle \in S$, where $0 \leq i \leq m$, and where there is no n'' such that $\langle a, n'', n_0 \rangle \in S$. Item (i) is true because T is saturated under rule RT. Item (ii) is true because T is saturated under rules RK and R4. To see that (iii) is true, note that $K_a\varphi_1$ is a sub-formula of φ_0 . Now, let n' be an a -antecedent of n . Then there is ψ such that $K_a\varphi_1$ is sub-formula of ψ , and such that $\langle \epsilon, n', \psi \rangle \in L$ (because n' is also a descendant of 0). Then either $\langle \epsilon, n', K_a\varphi_1 \rangle \in L$ or $\langle \epsilon, n', \neg K_a\varphi_1 \rangle \in L$ (by Lemma 16). In the first case, $\langle \epsilon, n', \varphi_1 \rangle \in L$ (because T is saturated under rule RT). And in the second case, $\langle \epsilon, n'', \neg K_a\varphi_1 \rangle \in L$ for all a -descendants of n' (because T is saturated under rule R5), which implies $\langle \epsilon, n, \neg K_a\varphi_1 \rangle \in L$. Then b is closed, which contradicts the hypothesis.

Let $\varphi = \neg K_a\varphi_1$. Then $\langle \epsilon, n', \neg\varphi_1 \rangle \in L$ for some n' , and $\langle a, n, n' \rangle \in S$ (because T is saturated under rule R \widehat{K}) iff $\langle M, n' \rangle \models \neg\varphi_1$ (by induction hypothesis) iff $\langle M, n' \rangle \models \neg K_a\varphi_1$.

Let $\varphi = [\varphi_1]\varphi_2$. Then $\{\langle \epsilon, n, \neg\varphi_1 \rangle\} \subseteq L$ or $\{\langle \epsilon, n, \varphi_1 \rangle, \langle (\varphi_1), n, \varphi_2 \rangle\} \subseteq L$ (because T is saturated under rule R $[\cdot]$) iff $\langle M, n \rangle \models \neg\varphi_1$ or $\langle M, n \rangle \models \varphi_1$ and $\langle M^{\varphi_1}, n \rangle \models \varphi_2$ (by induction hypothesis), iff $\langle M, n \rangle \models [\varphi_1]\varphi_2$.

Let $\varphi = \neg[\varphi_1]\varphi_2$. Then $\{\langle \epsilon, n, \varphi_1 \rangle, \langle (\varphi_1), n, \neg\varphi_2 \rangle\} \in L$ (because T is saturated under rule R $\langle \cdot \rangle$) iff $\langle M, n \rangle \models \varphi_1$ and $\langle M^{\varphi_1}, n \rangle \models \neg\varphi_2$ (by induction hypothesis), iff $\langle M, n \rangle \models \neg[\varphi_1]\varphi_2$.

Now, for the induction step, suppose $\lambda = \langle (\psi_1, \dots, \psi_k, \psi_{k+1}), n, \varphi \rangle$. We show $\mathcal{P}(\lambda)$, again by induction on the structure of φ . As before, there are two base cases. First, suppose that $\varphi = p$ for some $p \in P$. If $\lambda \in L$ then the labelled formulas $\langle \epsilon, n, \psi_1 \rangle, \langle (\psi_1), n, \psi_2 \rangle, \dots, \langle (\psi_1, \dots, \psi_k), n, \psi_{k+1} \rangle$, and $\langle (\psi_1, \dots, \psi_{k+1}), n, p \rangle$ are in L (by Lemma 15). Then $\langle M, n \rangle$ satisfies all the first k formulas above by induction hypothesis. Moreover, $\langle \epsilon, n, p \rangle \in L$ (because T is saturated under rule RSB), iff $n \in V_p$ iff $n \in V_p^{(\psi_1, \dots, \psi_{k+1})}$ and therefore $\langle M^{(\psi_1, \dots, \psi_{k+1})}, n \rangle \models p$. The argument is analogous for $\varphi = \neg p$.

The induction step is shown by induction on the structure of φ analogously to the case above ($k = 0$), with the difference that Lemma 15 is also used. We omit the details here. ■

4 Decision procedures for PAL

In this section we present an implementation of the tableau calculus defined in Section 3. We aim at showing that this implementation works in deterministic polynomial space, and thus conclude that the proposed proof method is optimal. Unfortunately, the implementation is, in its full details, long and hard to understand. Therefore, we first present the main idea and argue for its adequacy and termination in an informal way. This is done in Subsection 4.1. The actual algorithm is presented later, in Subsection 4.2.

4.1 Tableau strategies

A tableau strategy is a special way to apply the tableau rules. It is meant to prevent unnecessary rule applications and hence to guarantee termination and

optimality. From now on we call a labelled formula λ a *witness* to a given tableau rule in a branch $\langle L, S \rangle$ whenever:

- the labelled formula, alone, triggers the given rule (e.g., $\langle p, 0, \neg p \rangle$ triggers the rule RSB), or there exists $\sigma \in S$ such that the pair (λ, σ) triggers the given rule (e.g., the pair $(\langle \epsilon, 0, K_a p \rangle, \langle a, 0, 1 \rangle)$ triggers the rule RK); and
- λ is not marked as “non-applicable”. (In the strategy, a labelled formula is marked as “non-applicable” if the corresponding tableau rule has already been applied to it.)

STRATEGY 17

Take the initial tableau for φ_0 as input and execute the following steps.

1. If all branches in the latest generated tableau are closed, then halt and return “unsatisfiable”. Else if all branches in the latest generated tableau are either closed or saturated, then halt and return “satisfiable”. Else pick an open and non-saturated branch of the latest generated tableau and perform step 2.
2. If the branch contains a witness to one of the rules Rcut1, Rcut2, RSB, $R\neg$, $R\wedge$, $R\vee$, RT, $R[\cdot]$ and $R\langle \cdot \rangle$, then apply the corresponding rule, mark the witness as “non-applicable” in the new generated tableau, and then perform step 1. Else perform step 3.
3. If there is no witness to the rule $R\widehat{K}$, then mark this branch as “saturated” and then perform step 1, else perform the *inclusion test* (to be explained in the sequel) with the witness. If the inclusion test succeeds, then mark the witness as “non-applicable”, and then perform step 3 again, else apply rule $R\widehat{K}$, mark the witness as “non-applicable” in the new generated tableau, and then perform step 4.
4. Pick the branch corresponding to the current one in the new generated tableau. If the branch is not saturated under one of the rules RK, R4 and R5, then apply the corresponding rule, and then perform step 4 again. Else perform step 1.

As usual, the argument showing termination uses the fact that some rules are applied only once to the same witness, and also that the labelled formulas added to the branch by such rules are shorter than their witnesses.² It is the case for the rules applied in step 2. The rules applied in steps 3 and 4 however, do not have such property. This is why the *inclusion test* is essential to avoid infinite loops. This test is performed as follows.

DEFINITION 18 (INCLUSION TEST)

Let $\lambda = \langle (\psi_1, \dots, \psi_k), n, \neg K_a \varphi \rangle$ be the witness for $R\widehat{K}$ in the branch $\langle L, S \rangle$.

²The length of a labelled formula is the number of symbols occurring in it. $[\cdot]$ and K_a count two symbols each, and a list of the form (ψ_1, \dots, ψ_k) counts $\sum_{i \leq k} 1 + \text{len}(\psi_i)$ symbols. For example $\langle \epsilon, 0, \neg[q]K_a p \rangle$ has length 8, while $\langle (q), 0, \neg K_a p \rangle$ has length 7.

i. Generate the sets L' and L'' as follows:

$$\begin{aligned} L' &= \{\langle \epsilon, \neg[\psi_1] \dots [\psi_k] \varphi \rangle\} \\ L'' &= \{\langle \epsilon, [\psi_1] \dots [\psi_k] K_a \varphi' \rangle \mid \langle (\psi_1, \dots, \psi_k), n, K_a \varphi' \rangle \in L\} \cup \\ &\quad \{\langle \epsilon, [\psi_1] \dots [\psi_k] \neg K_a \varphi'' \rangle \mid \langle (\psi_1, \dots, \psi_k), n, \neg K_a \varphi'' \rangle \in L\}. \end{aligned}$$

And then perform step ii.

ii. If $L' \cup L'' \subseteq L_{n_0}$, where $L_{n_0} = \{\langle \mu, \psi \rangle \mid \langle \mu, n_0, \psi \rangle \in L\}$ and such that $\langle a, n_0, n_1 \rangle, \dots, \langle a, n_{i-1}, n_i \rangle \in S$, for some $i \leq 1$, and $n_i = n$, then the inclusion test *succeeds*, else it *fails*.

In step i, L' is filled with the formula that would be added to the branch as result of the application of rule $R\widehat{K}$, and L'' is filled with the formulas that would be added to the branch as result of the application of rules RK, R4 and R5. This means that the set $L' \cup L''$ contains the formulas to be added in what corresponds to an a -successor of the possible world n in the model under construction. Before the creation of this new world though, the strategy performs the inclusion test to verify whether its labelled formulas are already present in some a -predecessor world n_0 . For supposing that the latter happens, the new world would turn out to be inconsistent only if n_0 is inconsistent too. Moreover, the new world would have successors that can also be generated directly from n_0 . In this case, the new world does not need to be created. Therefore, it shows that the method remains complete with the inclusion test.

Before showing that it is enough to guarantee termination, we show through an example why it is necessary. First, let us call a branch *locally saturated* if and only if it is saturated under all tableau rules but $R\widehat{K}$. It should be clear that step 3 is executed only if the branch is locally saturated. Now, suppose that the strategy picks an open locally saturated branch containing, as witness to $R\widehat{K}$, the labelled formula $\langle \epsilon, n, \neg K_a \varphi \rangle$. Then step 3 is executed. Suppose that the inclusion test fails. Then rule $R\widehat{K}$ is applied to the witness. It generates a new tableau with a branch containing the labelled formula $\langle \epsilon, n', \neg \varphi \rangle$, and a skeleton containing $\langle a, n, n' \rangle$. Thus the latter branch is not saturated under R5. This branch is picked in step 4. The strategy will eventually apply R5, generating a tableau containing the labelled formula $\langle \epsilon, n', \neg K_a \varphi \rangle$. Note that the latter is a witness to $R\widehat{K}$, exactly as in the point we started. Therefore, the only thing that can avoid an infinite loop is the fact that the inclusion test will eventually succeed. For in the latter case, rule $R\widehat{K}$ is not applied to its witness.

In the proof of the theorem below (and further) we call a witness to $R\widehat{K}$ an *a-witness* whenever it is of the form $\langle \mu, n, \neg K_a \varphi \rangle$. That is, an *a-witness* to $R\widehat{K}$ is a witness to this rule such that its formula has the agent a as subscript of the outermost operator K .

THEOREM 19 (TERMINATION)

Strategy 17 halts for every input of the form $\{\{\langle \epsilon, 0, \varphi_0 \rangle\}, \emptyset\}$.

PROOF (SKETCH). It is enough to show that the strategy eventually generates a tableau such that all branches are either closed or saturated. For simplicity we do so in two steps. First, we take the hypothesis that the strategy never generates a branch containing a witness to the rule $R\widehat{K}$, and then show that the

strategy eventually generates a tableau such that all branches are either closed or saturated. Second, we show that for every input, the strategy eventually reaches a state such that no witness to the rule \widehat{RK} is ever generated.

Suppose that no witness to the rule \widehat{RK} is ever generated. Then, step 4 is never executed. Moreover, whenever step 3 is executed, it marks the current branch as “satisfiable”. Now, suppose that, during the execution of the strategy, the latest generated tableau T' contains a branch with a witness to one of the rules in step 2. Then, by executing step 2, the strategy generates a new tableau T wherein that witness is marked “non-applicable”, so it will never be a witness again. Moreover, T differs from T' by some additional labelled formulas that are shorter than that witness. Therefore, by a straightforward induction on the length of labelled formulas we show that the strategy will eventually generate a tableau such that all branches are either closed or saturated. The details are omitted.

To show that the strategy eventually reaches a state such that no witness to \widehat{RK} is ever generated, we also consider two cases. First, we take the hypothesis that the strategy generates only a -witnesses to \widehat{RK} , for some agent a . Note that in this case, every execution of step i of the inclusion test generates the same set L'' . To see that it is true, remember that at this stage the current branch is locally saturated. In particular, the current branch is saturated under the bounded cuts. This means that the second execution of the inclusion test with the same witness in the same branch always succeeds. During the whole process the strategy generates a finite number of different witnesses for rule \widehat{RK} , since the number of sub-formulas of φ_0 is finite. Then, after a finite number of executions of the inclusion test in the same branch, there will not be witnesses to \widehat{RK} any more.

In the second case, without loss of generality, we suppose that the strategy generates both a -witnesses and b -witnesses to rule \widehat{RK} . Now, suppose that, during the execution of the strategy, there is an a -witness λ to the rule \widehat{RK} of the form $\langle (\psi_1, \dots, \psi_k), n, \neg K_a \varphi \rangle$. Moreover suppose that the inclusion test with λ fails. Then after successive executions of step 4, when the strategy saturates the branch under rules RK , $R4$ and $R5$, all the labelled formulas added to the successor of n are either of the form $\langle \epsilon, n', \neg[\psi_1] \dots [\psi_k] \varphi' \rangle$, $\langle \epsilon, n', [\psi_1] \dots [\psi_k] \varphi' \rangle$, $\langle \epsilon, n', [\psi_1] \dots [\psi_k] K_a \varphi' \rangle$ or $\langle \epsilon, n', [\psi_1] \dots [\psi_k] \neg K_a \varphi' \rangle$, where n' is the a -successor of n . This means that any b -witness to \widehat{RK} that contains n' in its label is shorter than λ . Indeed, because they are sub-formulas of φ' . It follows that whenever an application of \widehat{RK} to an a -witness is followed by an application of \widehat{RK} to a b -witness, the length of every a -witnesses to \widehat{RK} present in the latest successor world is decreased. Therefore, it can be shown, by an (omitted) induction on the length of labelled formulas, that the strategy eventually reaches a state where all the possible witnesses to \widehat{RK} are either all of them a -witnesses, or all of them b -witnesses. And then, the first case above applies. ■

In order to have an optimal implementation of our method, we need to refine the above strategy. The refinements are meant to save memory. Instead of rebuilding the entire strategy, we presented the refinements as guidelines for the implementation. We recall that the implementation of this strategy is presented in full details in Section 4.2.

STRATEGY 20

Execute the same steps as Strategy 17 but taking into account the following guidelines.

- Work with only one tableau at a time, as follows. When applying a rule to the tableau T' , instead of keeping both T' and the new generated tableau T in the memory, erase the old tableau T' , and work with T only.
- Work with only one tableau branch at a time, as follows. If, in its definition, the application of a tableau rule generates more than one branch, then does not generate all the branches at once. Instead, mark this point as a “disjunctive branching point”, generate only one of the branches, and then perform a recursive call of the procedure for the new generated branch. If the recursive call returns “satisfiable”, then return “satisfiable”, else erase the first branch, generate the second one, perform a recursive call of the procedure for it, and so on. If there is no more branches to be generated, then return “unsatisfiable”.
- Work with only one possible world at a time, as follows. If there is more than one witness to the rule $R\hat{K}$, then mark this point as a “conjunctive branching point”, generate a tableau branch using only one of the witnesses, i.e., generate a tableau branch containing labelled formulas only for the formulas in $L' \cup L''$ of step i of the inclusion test, and then perform a recursive call of the procedure for the new generated tableau branch. If the recursive call returns “unsatisfiable”, then return “unsatisfiable”, else erase the first branch, generate the second one using the second witness, and perform a recursive call of the procedure for it, and so on. If there is no more witnesses to the rule $R\hat{K}$, then return “satisfiable”.

Note that the execution of Strategy 20 is a depth-first exploration of an and-or tree, much like the ‘tableau construction’ of Halpern and Moses (1992), that inspired this strategy. The nodes of the tree are sets of labelled formulas. And in each of these nodes all formulas are labelled by the same possible world. The disjunctive branchings are created by application of the rules R_{cut1} , R_{cut2} , R_V and $R[\cdot]$, and conjunctive branchings are created whenever a locally saturated node contains more than one witness to the rule $R\hat{K}$.

THEOREM 21 (COMPLEXITY)

The amount of memory used by Strategy 20 is a polynomial function on the length of φ_0 .

PROOF (SKETCH). The amount of memory used is limited by the maximum depth of the tree times the maximum size of its nodes.

The size of each node is at most $\mathcal{O}(16 \times \text{len}(\varphi_0)^4)$. To see it, note that the formulas in each node are all labelled by the same possible world. Then the size of each node is limited by the maximum number of different tuples of the form $\langle x, \mu, \varphi \rangle$ times the maximum length of each tuple, where:

- φ is a sub-formula or a negation of a sub-formula of φ_0 ;
- μ is the list of formulas in the label of φ ; and

- x is the mark of φ (We use marks to control whether a rule has already been applied to the corresponding formula, e.g., we store the mark “non-applicable” in x).

The length of each tuple is at most $\mathcal{O}(\text{len}(\varphi_0))$, and there are at most $16 \times \text{len}(\varphi_0)^3$ such tuples. To see that the latter is true note that: φ_0 has at most $2 \times \text{len}(\varphi_0)$ different sub-formulas and negations of sub-formulas; Strategy 20 generates at most $2 \times \text{len}(\varphi_0)^2$ different lists μ , because the longest list μ has at most $2 \times \text{len}(\varphi_0)$ sub-formulas or negations of sub-formulas of φ_0 , and because they always appear in μ in the same order as they appear as announcements in φ_0 ; and there are 8 different marks. (The actual implementation needs some more marks, for more details please see the implementation in Subsection 4.2.)

The maximum depth of the tree is $\mathcal{O}(\text{len}(\varphi_0)^7)$. First, Strategy 20 generates at most $\mathcal{O}(\text{len}(\varphi_0)^3)$ nodes before generating a node that is either closed or locally saturated. The latter is true because the number of labelled formulas in such a node is limited by the maximum number of different tuples of the form $\langle x, \mu, \varphi \rangle$. Second, the number of subsequent applications of rule $\widehat{\text{RK}}$ to a -witnesses, for some agent a , is limited by $\mathcal{O}(\text{len}(\varphi_0)^3)$. Again, because the number of different a -witnesses is limited by the maximum number of different tuples of the form $\langle x, \mu, \varphi \rangle$. Third, the application of rule $\widehat{\text{RK}}$ to a a -witness that follows an application of rule $\widehat{\text{RK}}$ to a b -witness, where a and b are any different agents, is limited by $\text{len}(\varphi_0)$. Remember that in the proof of Theorem 19 we saw that when an application of $\widehat{\text{RK}}$ to a a -witness follows an application of $\widehat{\text{RK}}$ to a b -witness, the lengths of the subsequent a -witnesses decrease. And the length of the very first witness is limited by $\text{len}(\varphi_0)$.

All together implies that the amount of memory used by Strategy 20 is a polynomial function on the length of φ_0 . ■

Therefore, the computational complexity of Strategy 20 is optimal, since satisfiability checking for public announcement logic is PSPACE-hard, given the fact that it is so for S5_n (Halpern and Moses, 1992).

4.2 Algorithm

In this section we present an algorithm that implements Strategy 20. For clarity, it is presented in four parts, but the reader should keep in mind that it consists in a single procedure.

The first part is the main procedure `SatPal`, presented in Figure 2. It takes a formula $\varphi_0 \in \mathcal{L}_{\text{PAL}}$ as input and returns ‘true’ if it is satisfiable, else it returns ‘false’. Lines 2–5 initialize the process. Variable $r_{0,0}$ contains the result of the computation. In the end of the execution it contains the value ‘true’ if φ_0 is satisfiable, and the value ‘false’ if φ_0 is not satisfiable. Variables i and j keep track of the branching points: i is incremented whenever the rule $\widehat{\text{RK}}$ is applied. That is, i counts the number of different possible worlds in a branch, and j is incremented whenever the tableau branches by the application of one of the rules `Rcut1`, `Rcut2`, `R \vee` and `R[.]`. Then $i + j$ is the number of branching points left behind. We also make memory allocations somehow explicit by the command ‘allocate’ (while memory freeing done in the sub-procedures is made explicit by the command ‘free’). It gives a relatively precise idea of how much memory is used. In line 4 the variables $L_{1,1}$ and $r_{1,1}$ are created. The

```

1: procedure SatPal input  $\varphi_0 \in \mathcal{L}_{\text{PAL}}$ 
2:    $r_{0,0} := \top$ 
3:    $i := 1, j := 1$ 
4:   allocate  $L_{i,j}, r_{i,j}$ 
5:    $L_{i,j} := \{\langle \text{new}, \epsilon, \varphi_0 \rangle\}$ 
6:   while  $i > 0$  do
7:     if  $L_{i,j}$  is closed then
8:        $r_{i,j} := \perp$ 
9:       Backtrack
10:    else
11:      Cut
12:      SaturateWorld
13:      CreateNewWorld
14:    end if
15:  end do
16:  return  $r_{0,0}$ 
17: end procedure

```

Figure 2: Procedure SatPal.

former implements the single branch of the initial tableau for φ_0 . The symbol ‘new’ indicates that no tableau rule has been applied to this labelled formula. Note that the triples do not contain the number n corresponding to its possible world. It is done in such a way because, following the idea of Halpern and Moses (1992), we explore only one possible world at a time, and therefore do not need to add this information to each labelled formula. For the same reason, we do not need a skeleton. For simplicity, we abuse the language and also call these triples labelled formulas. The variable $r_{i,j}$ contains the result of the computation for the branch $L_{i,j}$. Lines 7–9 stop the expansion of the branch $L_{i,j}$ if it is closed, and in this case, the value of $r_{i,j}$ is set accordingly. In the subsequent lines, we can find the in line sub-procedures that operate on the branches by adding and removing labelled formulas. They implement the tableau rules given in Definition 8 in an efficient way.

The sub-procedure Backtrack implements backtracking on branches, and is also part of the sub-procedure CreateNewWorld. It is more technical than interesting and is conveniently presented in the end. The sub-procedure Cut implements rules Rcut1 and Rcut2. It is cumbersome and does not add much to the understanding of the algorithm. Therefore we prefer to skip it.³ The sub-procedure SaturateWorld consists of an important component of the algorithm and is presented in Figure 3.

Lines 2–3, 4–5, 6–7, 8–9 and 10–11 of SaturateWorld implement respectively the rules RSB, $R\neg$, $R\wedge$, RT and $R\langle \cdot \rangle$. The symbol ‘cut’ means that the respective cut-rule (either Rcut1 or Rcut2) has already been applied to the corresponding labelled formula, while the symbol ‘app’ means that all applicable rules have already been applied to the corresponding labelled formula. Lines 12–22 implement rule $R\vee$. There are two else-if parts: the first one (lines 12–17) corresponds to the exploration of the right most branch, i.e., the branch containing

³The interested reader can find it in Appendix B.

```

1: inline procedure SaturateWorld
2:   else if  $\langle \text{new}, \mu \bullet (\psi), \ell \rangle \in L_{i,j}$ , for some  $\mu, \psi$  and some literal  $\ell$  then
3:      $L_{i,j} := L_{i,j} \setminus \{ \langle \text{app}, \mu \bullet (\psi), \ell \rangle \} \cup \{ \langle \text{new}, \mu, \ell \rangle \}$ 
4:   else if  $\langle \text{new}, \mu, \neg \neg \varphi \rangle \in L_{i,j}$ , for some  $\mu, \varphi$  then
5:      $L_{i,j} := L_{i,j} \setminus \{ \langle \text{new}, \mu, \neg \neg \varphi \rangle \} \cup \{ \langle \text{app}, \mu, \neg \neg \varphi \rangle, \langle \text{new}, \mu, \varphi \rangle \}$ 
6:   else if  $\langle \text{new}, \mu, \varphi_1 \wedge \varphi_2 \rangle \in L_{i,j}$ , for some  $\mu, \varphi_1, \varphi_2$  then
7:      $L_{i,j} := L_{i,j} \setminus \{ \langle \text{new}, \mu, \varphi_1 \wedge \varphi_2 \rangle \} \cup$ 
            $\{ \langle \text{app}, \mu, \varphi_1 \wedge \varphi_2 \rangle, \langle \text{new}, \mu, \varphi_1 \rangle, \langle \text{new}, \mu, \varphi_2 \rangle \}$ 
8:   else if  $\langle \text{cut}, \mu, K_a \varphi \rangle \in L_{i,j}$ , for some  $a, \mu, \varphi$  then
9:      $L_{i,j} := L_{i,j} \setminus \{ \langle \text{cut}, \mu, K_a \varphi \rangle \} \cup \{ \langle \text{app}, \mu, K_a \varphi \rangle, \langle \text{new}, \mu, \varphi \rangle \}$ 
10:  else if  $\langle \text{cut}, \mu, \neg[\varphi_1]\varphi_2 \rangle \in L_{i,j}$ , for some  $\mu, \varphi_1, \varphi_2$  then
11:     $L_{i,j} := L_{i,j} \setminus \{ \langle \text{cut}, \mu, \neg[\varphi_1]\varphi_2 \rangle \} \cup$ 
            $\{ \langle \text{app}, \mu, \neg[\varphi_1]\varphi_2 \rangle, \langle \text{new}, \mu, \varphi_1 \rangle, \langle \text{new}, \mu \bullet (\varphi_1), \varphi_2 \rangle \}$ 
12:  else if  $\langle \text{cut}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L_{i,j}$ , for some  $\mu, \varphi_1, \varphi_2$  then
13:     $L_{i,j} := L_{i,j} \setminus \{ \langle \text{cut}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \} \cup \{ \langle \text{a1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \}$ 
14:     $r_{i,j} := \perp$ 
15:     $j := j + 1$ 
16:    allocate  $L_{i,j}, r_{i,j}$ 
17:     $L_{i,j} := L_{i,j-1} \setminus \{ \langle \text{a1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \} \cup \{ \langle \text{app}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle, \langle \text{new}, \mu, \neg \varphi_1 \rangle \}$ 
18:  else if  $\langle \text{a1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L_{i,j}$ , for some  $\mu, \varphi_1, \varphi_2$  then
19:     $L_{i,j} := L_{i,j} \setminus \{ \langle \text{a1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \} \cup \{ \langle \text{app}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \}$ 
20:     $j := j + 1$ 
21:    allocate  $L_{i,j}, r_{i,j}$ 
22:     $L_{i,j} := L_{i,j-1} \cup \{ \langle \text{new}, \mu, \neg \varphi_2 \rangle \}$ 
23:  else if  $\langle \text{cut}, \mu, [\varphi_1]\varphi_2 \rangle \in L_{i,j}$  for some  $\mu, \varphi_1, \varphi_2$  then
24:     $L_{i,j} := L_{i,j} \setminus \{ \langle \text{cut}, \mu, [\varphi_1]\varphi_2 \rangle \} \cup \{ \langle \text{a1}, \mu, [\varphi_1]\varphi_2 \rangle \}$ 
25:     $r_{i,j} := \perp$ 
26:     $j := j + 1$ 
27:    allocate  $L_{i,j}, r_{i,j}$ 
28:     $L_{i,j} := L_{i,j-1} \setminus \{ \langle \text{a1}, \mu, [\varphi_1]\varphi_2 \rangle \} \cup \{ \langle \text{app}, \mu, [\varphi_1]\varphi_2 \rangle, \langle \text{new}, \mu, \neg \varphi_1 \rangle \}$ 
29:  else if  $\langle \text{a1}, \mu, [\varphi_1]\varphi_2 \rangle \in L_{i,j}$  for some  $\mu, \varphi_1, \varphi_2$  then
30:     $L_{i,j} := L_{i,j} \setminus \{ \langle \text{a1}, \mu, [\varphi_1]\varphi_2 \rangle \} \cup \{ \langle \text{app}, \mu, [\varphi_1]\varphi_2 \rangle \}$ 
31:     $j := j + 1$ 
32:    allocate  $L_{i,j}, r_{i,j}$ 
33:     $L_{i,j} := L_{i,j-1} \cup \{ \langle \text{new}, \mu, \varphi_1 \rangle, \langle \text{new}, \mu \bullet (\varphi_1), \varphi_2 \rangle \}$ 
34: end inline procedure

```

Figure 3: Sub-procedure SaturateWorld.

```

1: inline procedure CreateNewWorld
2:   else if  $\langle \text{cut}, (\psi_1, \dots, \psi_k), \neg K_a \varphi \rangle \in L_{i,j}$ , for some  $\psi_1, \dots, \psi_k, a, \varphi$  then
3:      $L' := \{ \langle \text{new}, \epsilon, \neg[\psi_1] \dots [\psi_k] \varphi \rangle \}$ 
4:      $L'' := \{ \langle \text{new}, \epsilon, [\psi'_1] \dots [\psi'_k] K_a \varphi' \rangle \mid \langle y, (\psi'_1, \dots, \psi'_k), K_a \varphi' \rangle \in L_{i,j} \} \cup$ 
5:        $\{ \langle \text{new}, \epsilon, [\psi''_1] \dots [\psi''_k] \neg K_a \varphi'' \rangle \mid \langle z, (\psi''_1, \dots, \psi''_k), \neg K_a \varphi'' \rangle \in L_{i,j} \}$ 
6:      $L_{i,j} := L_{i,j} \setminus \{ \langle \text{cut}, (\psi_1, \dots, \psi_k), \neg K_a \varphi \rangle \} \cup \{ \langle \text{app}, (\psi_1, \dots, \psi_k), \neg K_a \varphi \rangle \}$ 
7:     if  $L' \cup L'' \not\subseteq L_{k,1}$  for some  $1 \leq k \leq i$  such that  $k < i$  implies
8:        $a = s_{i-1} = \dots = s_{i-m} = s_k$  for some  $0 \leq m \leq i - k$  then
9:          $r_{i,j} := \top$ 
10:        allocate  $d_i, d_i := j$ 
11:         $i := i + 1, j := 1$ 
12:        allocate  $L_{i,j}, r_{i,j}, s_i$ 
13:         $L_{i,j} := L' \cup L'', s_i := a$ 
14:     end if
15:   else
16:      $r_{i,j} := \top$ 
17:     Backtrack
18:   end if
19: end inline procedure

```

Figure 4: Sub-procedure CreateNewWorld.

the formula $\neg\varphi_1$; the second one (lines 18–22) corresponds to the exploration of the left most branch, i.e., the branch containing the formula $\neg\varphi_2$. The symbol ‘a1’ is used in line 13 to indicate that the rule $R\vee$ has been applied once and the program is currently exploring the right most branch. The value ‘false’ is assigned to the variable $r_{i,j}$ in line 14 because this point is a disjunctive branching point (it will become clearer in the explanation of the sub-procedure Backtrack), i.e., its result depends on the disjunction of the results of its children. The variable j is incremented in line 15, a new branch is created in line 16, and the a new labelled formula containing $\neg\varphi_1$ is added to the new branch in line 17. After the exploration of the right most branch the program eventually goes back to this branching point and finds, in branch $L_{i,j}$, the labelled formula $\langle \text{a1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle$. Then it explores the left most branch by executing the second else-if part. It replaces, in line 19, the symbol ‘a1’ for the symbol ‘app’ to indicate that all applicable rules ($R\vee$ in this case) have already been applied to that labelled formula. Analogously to the first else-if part, the algorithm moreover creates a new branch and adds a new labelled formula containing $\neg\varphi_2$ to it. Lines 23–33 implement rule $R[\cdot]$ in a similar way.

Note that for each tableau rule implemented by procedure SaturateWorld, it adds to the tableau labelled formulas that are shorter than the formula that triggered the rule. Rules $R\widehat{K}$, RK , $R4$ and $R5$ however, do not present such property. Therefore, to obtain termination of SatPal, their implementation has a loop test mechanism. All this is done in procedure CreateNewWorld, presented in Figure 4.

In line 3 of CreateNewWorld, L' is filled with the formula resulting from application of rule $R\widehat{K}$. In line 4, L'' is filled with all the formulas resulting from the application of rules RK , $R4$ and $R5$. Together, L' and L'' contain all

```

1: inline procedure Backtrack
2:    $j := j - 1;$ 
3:   if  $j \geq 1$  then
4:      $r_{i,j} := r_{i,j}$  or  $r_{i,j+1};$ 
5:     free  $L_{i,j+1}, r_{i,j+1}$ 
6:   else
7:      $i := i - 1$ 
8:     if  $i > 0$  then
9:        $j := d_{i+1}$ 
10:      free  $s_{i+1}$ 
11:    end if
12:     $r_{i,j} := r_{i,j}$  and  $r_{i+1,1};$ 
13:    free  $d_{i+1}$ 
14:    free  $L_{i+1,1}, r_{i+1,1}$ 
15:  end if
16: end inline procedure

```

Figure 5: Sub-procedure Backtrack.

the formulas to be put in a new branch corresponding to a new world in the model. But before the creation of this branch, in line 6, the procedure verifies whether this branch would not be contained in one of its predecessors $L_{k,1}$, for some $k \leq i$. For supposing that $L_{k,1}$ contains $L' \cup L''$, the new branch would have successors that can also be generated directly from $L_{k,1}$. Moreover, the new branch would be closed only if $L_{k,1}$ is closed too. Therefore, the new branch do not need to be created. In the case that no predecessor contains $L' \cup L''$, a new branch is created. The value 'true' is assigned to the variable $r_{i,j}$ in line 7 because this is a conjunctive branching point, i.e., its result depends on a conjunction of the results of its successors. Variable d_i is created only to keep the ancient value of j , that is set to 1 just after the increment of i . A new branch $L_{i,1}$ is created in line 10 together with the variable s_i . The latter contains the label of the link between $L_{i,j}$ and its predecessor in the model.

Figure 5 presents sub-procedure Backtrack. As said before, it is more technical than interesting. It decrements j and checks whether the current branch was created in a disjunctive branching point, i.e., it was created by the application of one of the rules Rcut1, Rcut2, R \vee or R[·]. If it is the case, then the result of its predecessor ($r_{i,j}$) is a disjunction of the results of all its successors. Else the current branch was created in a conjunctive branching point, i.e., it was created by the application of the rule R \widehat{K} . Then it decrements i and compute the result of its predecessor as a conjunction of the results of its children. Backtrack moreover restores the value of j and free memory that is not used anymore.

5 Variants of PAL

In earlier work (Balbiani et al., 2007b) we considered some public announcement logics for which the underlying epistemic logic is not $S5_n$. There we call K-PAL the logic for which no restriction is imposed to the relations R_a , and

we call **KT-PAL** and **S4-PAL** the logics for which the relations R_a are respectively reflexive, and reflexive and transitive. In that work we moreover present strategies that lead to optimal implementations of the method for **K-PAL** and **KT-PAL**. Note however that the method presented here subsumes the old one, since simple adaptations transform it into a tableau method for each one of these logics. Namely, without the rule **R5**, one clearly obtains a sound and complete tableau method for **S4-PAL**. If moreover the rule **R4** is removed, one obtains a method for **KT-PAL**, and moreover removing the rule **RT**, a method for **K-PAL** is obtained. If the very same adaptations were done in Strategy 20 and the algorithm of Subsection 4.2, then one would obtain optimal decision procedures for each of these logics. In addition, note that albeit sound, the rules **Rcut1** and **Rcut2** are not necessary for logics without the axiom 5. It means that the method is also cut-free for **K-PAL**, **KT-PAL** and **S4-PAL**.

However, proofs of termination and complexity of a cut-free strategy for **S4-PAL** must take an additional detail into account. Let us recall the argument used in the proof of Theorem 19 in the point where we suppose that the strategy generates only a -witnesses to $R\widehat{K}$, for some agent a . Without cut rules, it is no longer the case that the set L'' generated in step i of the inclusion test is always the same. Here instead, we use the fact that every execution of step i in the same branch generates a set L'' that contains the set L'' generated in the previous execution of this step. This means that after at most $16 \times \text{len}(\varphi_0)^3$ executions of step i in the same branch, further executions of this step generate identical sets L'' each time. It is the case because this is the maximal number of different tuples of the form $\langle x, \mu, \varphi \rangle$ (cf. first part of the proof of Theorem 19). The rest of the argument is the same. Therefore the cut-free strategy for **S4-PAL** terminates. Moreover, it follows that the cut-free strategy for **S4-PAL** uses more memory, but this is still limited by a polynomial function on the length of φ_0 .

6 Conclusion

In this article we propose a proof method for **PAL** using analytic tableaux. Recently, a different tableau method for **PAL** has been developed independently by de Boer (2007). Apart from some aesthetic differences, his method is similar to ours. There, each formula is also prefixed by a label formed by a pair representing a sequence of updates and a possible world. De Boer's tableau rules are simpler and more elegant than ours, but less modular. It seems that his tableau rules do not permit, as easily as ours, the implementation of tableau methods for **K-PAL**, **KT-PAL**, nor **S4-PAL**. De Boer also argues that his tableau system can simulate our previous method (i.e., the method proposed in (Balbiani et al., 2007b)), but unfortunately do not provide proofs of decidability nor computational complexity.

The present method is quite flexible and maybe the reader is now wondering why we do not consider a public announcement logic whose the underlying epistemic logic is the one commonly used to model agents' beliefs: $KD45_n$. That is, the logic for which models are serial, transitive and euclidean. Seriality corresponds to the axiom scheme D . $K_a\varphi \rightarrow \neg K_a\neg\varphi$, and could easily be encoded into a tableau rule. However, the class of models where the accessibility relation is serial, transitive and euclidean is not closed for the model update

operation defined by public announcements.

As possible future works, we can explore the flexibility of our method to consider some “natural” extensions of public announcement logic. PAL with common knowledge and PAL with relativized common knowledge are good examples. The idea of making explicit the model updates in the labels of the tableau seems promising for addressing other dynamic epistemic logics too. For instance, (and exactly as done in (Balbiani et al., 2007b)) a proof method for the recently proposed arbitrary public announcement logic (Balbiani et al., 2007a) can be obtained through a straightforward extension. Another example of possible future work is PAL with public assignments (van Ditmarsch et al., 2005). Without going into deep details, we just mention here that its proof system is also built up with reduction axioms.

Actually, proof systems for all DELs are usually built up with such axioms. They can be used to eliminate the dynamic operators and thus rewrite the formulas in a simpler language. However, the price is an exponentially larger formula to be evaluated in the worst case. As proved here and also in (Lutz, 2006), this price is not mandatory for PAL. Therefore, one of the questions raised by this article is whether direct methods, similar to the presented one, can also be applied to other DELs.

Acknowledgments

First, we thank Olivier Gasquet for a discussion that helped us to find the optimal implementation of the tableau method. Second, we also thank the two anonymous reviewers for comments that helped us to substantially improve this contribution.

References

- Philippe Balbiani, Alexandru Baltag, Hans van Ditmarsch, Andreas Herzig, Tomohiro Hoshi, and Tiago de Lima. What can we achieve by arbitrary announcements? A dynamic take on Fitch’s knowability. In Dov Samet, editor, *Proceedings of the Eleventh Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 42–51. Presses universitaires de Louvain, 2007a. ISBN 978-2-8746-3077-4.
- Philippe Balbiani, Hans van Ditmarsch, Andreas Herzig, and Tiago de Lima. A tableau method for public announcement logics. In N. Olivetti, editor, *Proceedings of the Sixteenth International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, volume 4548 of *Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence)*, pages 43–59. Springer-Verlag, 2007b.
- Alexandru Baltag, Lawrence Moss, and Slawomir Solecki. The logic of common knowledge, public announcements, and private suspicions. In *Proceedings of the Seventh Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 43–46. Morgan Kaufmann Publishers Inc., 1998.

- Mathijs de Boer. KE tableaux for public announcement logic. In *Proceedings of Formal Approaches to Multi-Agent Systems Workshop (FAMAS)*, 2007. URL <http://www.mimuw.edu.pl/MAS/FAMAS007/>.
- Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel Publishing Company, 1983.
- Jelle Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, ILLC, University of Amsterdam, 1999.
- Joseph Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:311–379, 1992.
- Barteld Kooi. Expressivity and completeness for public update logic via reduction axioms. *Journal of Applied Non-Classical Logics*, 17(2):231–253, 2007.
- Carsten Lutz. Complexity and succinctness of public announcement logic. In P. Stone and G. Weiss, editors, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 137–144, 2006.
- Jan Plaza. Logics of public communications. In M. L. Emrich, M. Hadzikadic, M. S. Pfeifer, and Z. W. Ras, editors, *Proceedings of the Fourth International Symposium on Methodologies for Intelligent Systems (ISMIS)*, pages 201–216, 1989.
- Johan van Benthem, Jan van Eijck, and Barteld Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.
- Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. Dynamic epistemic logic with assignment. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. Singh, and M. Wooldridge, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 141–148. ACM, 2005.

A Tableau rules in “nominator/denominator” form

$$\begin{array}{l}
 \text{Rcut1} \quad \frac{\langle \mu, n, \neg(\varphi_1 \wedge \varphi_2) \rangle}{\begin{array}{c|c|c|c} \langle \mu, n, \neg\varphi_1 \rangle & \langle \mu, n, \neg\varphi_1 \rangle & \langle \mu, n, \varphi_1 \rangle & \langle \mu, n, \varphi_1 \rangle \\ \langle \mu, n, \neg\varphi_2 \rangle & \langle \mu, n, \varphi_2 \rangle & \langle \mu, n, \neg\varphi_2 \rangle & \langle \mu, n, \varphi_2 \rangle \end{array}} \\
 \\
 \text{Rcut1}' \quad \frac{\langle \mu, n, [\varphi_1]\varphi_2 \rangle}{\begin{array}{c|c|c|c} \langle \mu, n, \neg\varphi_1 \rangle & \langle \mu, n, \neg\varphi_1 \rangle & \langle \mu, n, \varphi_1 \rangle & \langle \mu, n, \varphi_1 \rangle \\ \langle \mu, n, \neg\varphi_2 \rangle & \langle \mu, n, \varphi_2 \rangle & \langle \mu, n, \neg\varphi_2 \rangle & \langle \mu, n, \varphi_2 \rangle \end{array}} \\
 \\
 \text{Rcut2} \quad \frac{\langle \mu, n, \neg[\varphi_1]\varphi_2 \rangle}{\begin{array}{c|c} \langle \mu, n, \neg\varphi_2 \rangle & \langle \mu, n, \varphi_2 \rangle \end{array}} \\
 \\
 \text{Rcut2}' \quad \frac{\langle \mu, n, K_a\varphi_2 \rangle}{\begin{array}{c|c} \langle \mu, n, \neg\varphi_2 \rangle & \langle \mu, n, \varphi_2 \rangle \end{array}}
 \end{array}$$

$$\begin{array}{l}
\text{Rcut2''} \quad \frac{\langle \mu, n, \neg K_a \varphi_2 \rangle}{\langle \mu, n, \neg \varphi_2 \rangle \mid \langle \mu, n, \varphi_2 \rangle} \\
\text{RSB} \quad \frac{\langle \mu \bullet (\psi), n, \ell \rangle}{\langle \mu, n, \ell \rangle} \\
\text{R}\neg \quad \frac{\langle \mu, n, \neg \neg \varphi \rangle}{\langle \mu, n, \varphi \rangle} \\
\text{R}\wedge \quad \frac{\langle \mu, n, \varphi_1 \wedge \varphi_2 \rangle}{\langle \mu, n, \varphi_1 \rangle \quad \langle \mu, n, \varphi_2 \rangle} \\
\text{RV} \quad \frac{\langle \mu, n, \neg(\varphi_1 \wedge \varphi_2) \rangle}{\langle \mu, n, \neg \varphi_1 \rangle \mid \langle \mu, n, \neg \varphi_2 \rangle} \\
\text{RT} \quad \frac{\langle \mu, n, K_a \varphi \rangle}{\langle \mu, n, \varphi \rangle} \\
\text{RK} \quad \frac{\langle (\psi_1, \dots, \psi_k), n, K_a \varphi \rangle : \langle a, n, n' \rangle}{\langle \epsilon, n', [\psi_1] \dots [\psi_k] \varphi \rangle} \\
\text{R4} \quad \frac{\langle (\psi_1, \dots, \psi_k), n, K_a \varphi \rangle : \langle a, n, n' \rangle}{\langle \epsilon, n', [\psi_1] \dots [\psi_k] K_a \varphi \rangle} \\
\text{R5} \quad \frac{\langle (\psi_1, \dots, \psi_k), n, \neg K_a \varphi \rangle : \langle a, n, n' \rangle}{\langle \epsilon, n', [\psi_1] \dots [\psi_k] \neg K_a \varphi \rangle}
\end{array}$$

$$\text{R}\widehat{K} \quad \frac{\langle (\psi_1, \dots, \psi_k), n, \neg K_a \varphi \rangle}{\langle \epsilon, n', \neg [\psi_1] \dots [\psi_k] \varphi \rangle : \langle a, n, n' \rangle} \quad \text{for some } n' \in \mathbb{N} \text{ not occurring in the branch.}$$

$$\begin{array}{l}
\text{R}[\cdot] \quad \frac{\langle \mu, n, [\varphi_1] \varphi_2 \rangle}{\langle \mu, n, \neg \varphi_1 \rangle \mid \langle \mu, n, \varphi_1 \rangle} \\
\text{R}\langle \cdot \rangle \quad \frac{\langle \mu, n, \neg [\varphi_1] \varphi_2 \rangle}{\langle \mu, n, \varphi_1 \rangle \quad \langle \mu \bullet (\varphi_1), n, \varphi_2 \rangle}
\end{array}$$

B Sub-procedure Cut

- 1: **inline procedure** Cut
- 2: **else if** $\langle \text{new}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
- 3: $L_{i,j} := L_{i,j} \setminus \{ \langle \text{new}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \} \cup \{ \langle \text{c1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \}$
- 4: $r_{i,j} := \perp, j := j + 1$
- 5: **allocate** $L_{i,j}$
- 6: $L_{i,j} := L_{i,j-1} \setminus \{ \langle \text{c1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \} \cup \{ \langle \text{cut}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle, \langle \text{new}, \mu, \neg \varphi_1 \rangle, \langle \text{new}, \mu, \neg \varphi_2 \rangle \}$
- 7: **else if** $\langle \text{c1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
- 8: $L_{i,j} := L_{i,j} \setminus \{ \langle \text{c1}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \} \cup \{ \langle \text{c2}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \}$
- 9: $j := j + 1$

10: **allocate** $L_{i,j}$
11: $L_{i,j} := L_{i,j-1} \setminus \{\langle c2, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle\} \cup \{\langle \text{cut}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle, \langle \text{new}, \mu, \neg\varphi_1 \rangle, \langle \text{new}, \mu, \varphi_2 \rangle\}$
12: **else if** $\langle c2, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
13: $L_{i,j} := L_{i,j} \setminus \{\langle c2, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle\} \cup \{\langle c3, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle\}$
14: $j := j + 1$
15: **allocate** $L_{i,j}$
16: $L_{i,j} := L_{i,j-1} \setminus \{\langle c3, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle\} \cup \{\langle \text{cut}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle, \langle \text{new}, \mu, \varphi_1 \rangle, \langle \text{new}, \mu, \neg\varphi_2 \rangle\}$
17: **else if** $\langle c3, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
18: $L_{i,j} := L_{i,j} \setminus \{\langle c3, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle\} \cup \{\langle \text{cut}, \mu, \neg(\varphi_1 \wedge \varphi_2) \rangle\}$
19: $j := j + 1$
20: **allocate** $L_{i,j}$
21: $L_{i,j} := L_{i,j-1} \cup \{\langle \text{new}, \mu, \varphi_1 \rangle, \langle \text{new}, \mu, \varphi_2 \rangle\}$
22: **else if** $\langle \text{new}, \mu, \neg K_a \varphi \rangle \in L_{i,j}$, for some μ, a, φ **then**
23: $L_{i,j} := L_{i,j} \setminus \{\langle \text{new}, \mu, \neg K_a \varphi \rangle\} \cup \{\langle c1, \mu, \neg K_a \varphi \rangle\}$
24: $r_{i,j} := \perp, j := j + 1$
25: **allocate** $L_{i,j}$
26: $L_{i,j} := L_{i,j-1} \setminus \{\langle c1, \mu, \neg K_a \varphi \rangle\} \cup \{\langle \text{cut}, \mu, \neg K_a \varphi \rangle, \langle \text{new}, \mu, \neg\varphi \rangle\}$
27: **else if** $\langle c1, \mu, \neg K_a \varphi \rangle \in L_{i,j}$, for some μ, a, φ **then**
28: $L_{i,j} := L_{i,j} \setminus \{\langle c1, \mu, \neg K_a \varphi \rangle\} \cup \{\langle \text{cut}, \mu, \neg K_a \varphi \rangle\}$
29: $j := j + 1$
30: **allocate** $L_{i,j}$
31: $L_{i,j} := L_{i,j-1} \cup \{\langle \text{new}, \mu, \varphi \rangle\}$
32: **else if** $\langle \text{new}, \mu, [\varphi_1]\varphi_2 \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
33: $L_{i,j} := L_{i,j} \setminus \{\langle \text{new}, \mu, [\varphi_1]\varphi_2 \rangle\} \cup \{\langle c1, \mu, [\varphi_1]\varphi_2 \rangle\}$
34: $r_{i,j} := \perp, j := j + 1$
35: **allocate** $L_{i,j}$
36: $L_{i,j} := L_{i,j-1} \setminus \{\langle c1, \mu, [\varphi_1]\varphi_2 \rangle\} \cup \{\langle \text{cut}, \mu, [\varphi_1]\varphi_2 \rangle, \langle \text{new}, \mu, \neg\varphi_1 \rangle, \langle \text{new}, \mu, \neg\varphi_2 \rangle\}$
37: **else if** $\langle c1, \mu, [\varphi_1]\varphi_2 \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
38: $L_{i,j} := L_{i,j} \setminus \{\langle c1, \mu, [\varphi_1]\varphi_2 \rangle\} \cup \{\langle c2, \mu, [\varphi_1]\varphi_2 \rangle\}$
39: $j := j + 1$
40: **allocate** $L_{i,j}$
41: $L_{i,j} := L_{i,j-1} \setminus \{\langle c2, \mu, [\varphi_1]\varphi_2 \rangle\} \cup \{\langle \text{cut}, \mu, [\varphi_1]\varphi_2 \rangle, \langle \text{new}, \mu, \neg\varphi_1 \rangle, \langle \text{new}, \mu, \varphi_2 \rangle\}$
42: **else if** $\langle c2, \mu, [\varphi_1]\varphi_2 \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
43: $L_{i,j} := L_{i,j} \setminus \{\langle c2, \mu, [\varphi_1]\varphi_2 \rangle\} \cup \{\langle c3, \mu, [\varphi_1]\varphi_2 \rangle\}$
44: $j := j + 1$
45: **allocate** $L_{i,j}$
46: $L_{i,j} := L_{i,j-1} \setminus \{\langle c3, \mu, [\varphi_1]\varphi_2 \rangle\} \cup \{\langle \text{cut}, \mu, [\varphi_1]\varphi_2 \rangle, \langle \text{new}, \mu, \varphi_1 \rangle, \langle \text{new}, \mu, \neg\varphi_2 \rangle\}$
47: **else if** $\langle c3, \mu, [\varphi_1]\varphi_2 \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
48: $L_{i,j} := L_{i,j} \setminus \{\langle c3, \mu, [\varphi_1]\varphi_2 \rangle\} \cup \{\langle \text{cut}, \mu, [\varphi_1]\varphi_2 \rangle\}$
49: $j := j + 1$
50: **allocate** $L_{i,j}$
51: $L_{i,j} := L_{i,j-1} \cup \{\langle \text{new}, \mu, \varphi_1 \rangle, \langle \text{new}, \mu, \neg\varphi_2 \rangle\}$
52: **else if** $\langle \text{new}, \mu, \neg[\varphi_1]\varphi_2 \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
53: $L_{i,j} := L_{i,j} \setminus \{\langle \text{new}, \mu, \neg[\varphi_1]\varphi_2 \rangle\} \cup \{\langle c1, \mu, \neg[\varphi_1]\varphi_2 \rangle\}$
54: $r_{i,j} := \perp, j := j + 1$
55: **allocate** $L_{i,j}$
56: $L_{i,j} := L_{i,j-1} \setminus \{\langle c1, \mu, \neg[\varphi_1]\varphi_2 \rangle\} \cup \{\langle \text{cut}, \mu, \neg[\varphi_1]\varphi_2 \rangle, \langle \text{new}, \mu, \neg\varphi_2 \rangle\}$
57: **else if** $\langle c1, \mu, \neg[\varphi_1]\varphi_2 \rangle \in L_{i,j}$, for some $\mu, \varphi_1, \varphi_2$ **then**
58: $L_{i,j} := L_{i,j} \setminus \{\langle c1, \mu, \neg[\varphi_1]\varphi_2 \rangle\} \cup \{\langle \text{cut}, \mu, \neg[\varphi_1]\varphi_2 \rangle\}$
59: $j := j + 1$

```

60:   allocate  $L_{i,j}$ 
61:    $L_{i,j} := L_{i,j-1} \cup \{\langle \text{new}, \mu, \varphi_2 \rangle\}$ 
62:   else if  $\langle \text{new}, \mu, K_a \varphi \rangle \in L_{i,j}$ , for some  $\mu, \varphi$  then
63:      $L_{i,j} := L_{i,j} \setminus \{\langle \text{new}, \mu, K_a \varphi \rangle\} \cup \{\langle \text{c1}, \mu, K_a \varphi \rangle\}$ 
64:      $r_{i,j} := \perp, j := j + 1$ 
65:   allocate  $L_{i,j}$ 
66:    $L_{i,j} := L_{i,j-1} \setminus \{\langle \text{c1}, \mu, K_a \varphi \rangle\} \cup \{\langle \text{cut}, \mu, K_a \varphi \rangle, \langle \text{new}, \mu, \neg \varphi \rangle\}$ 
67:   else if  $\langle \text{c1}, \mu, K_a \varphi \rangle \in L_{i,j}$ , for some  $\mu, \varphi_1, \varphi$  then
68:      $L_{i,j} := L_{i,j} \setminus \{\langle \text{c1}, \mu, K_a \varphi \rangle\} \cup \{\langle \text{cut}, \mu, K_a \varphi \rangle\}$ 
69:      $j := j + 1$ 
70:   allocate  $L_{i,j}$ 
71:    $L_{i,j} := L_{i,j-1} \cup \{\langle \text{new}, \mu, \varphi \rangle\}$ 
72:   else if  $\langle \text{new}, \mu, \neg K_a \varphi \rangle \in L_{i,j}$ , for some  $\mu, \varphi$  then
73:      $L_{i,j} := L_{i,j} \setminus \{\langle \text{new}, \mu, \neg K_a \varphi \rangle\} \cup \{\langle \text{c1}, \mu, \neg K_a \varphi \rangle\}$ 
74:      $r_{i,j} := \perp, j := j + 1$ 
75:   allocate  $L_{i,j}$ 
76:    $L_{i,j} := L_{i,j-1} \setminus \{\langle \text{c1}, \mu, \neg K_a \varphi \rangle\} \cup \{\langle \text{cut}, \mu, \neg K_a \varphi \rangle, \langle \text{new}, \mu, \neg \varphi \rangle\}$ 
77:   else if  $\langle \text{c1}, \mu, \neg K_a \varphi \rangle \in L_{i,j}$ , for some  $\mu, \varphi_1, \varphi$  then
78:      $L_{i,j} := L_{i,j} \setminus \{\langle \text{c1}, \mu, \neg K_a \varphi \rangle\} \cup \{\langle \text{cut}, \mu, \neg K_a \varphi \rangle\}$ 
79:      $j := j + 1$ 
80:   allocate  $L_{i,j}$ 
81:    $L_{i,j} := L_{i,j-1} \cup \{\langle \text{new}, \mu, \varphi \rangle\}$ 
82: end inline procedure

```