Cut-free Single-pass Tableaux for the Logic of Common Knowledge

Pietro Abate[1], Rajeev Goré[1], and Florian Widmann[2]

[1] The Australian National University
Canberra ACT 0200, Australia
{Pietro.Abate|Rajeev.Gore}@anu.edu.au

[2] The Australian National University and NICTA*
Canberra ACT 0200, Australia
Florian.Widmann@anu.edu.au

Abstract. We present a cut-free tableau calculus with histories and variables for the EXPTIME-complete multi-modal logic of common knowledge (LCK). Our calculus constructs the tableau using only one pass, so proof-search for testing theoremhood of φ does not exhibit the worst-case EXPTIME-behaviour for *all* φ as in two-pass methods. Our calculus also does not contain a "finitized ω-rule" so that it detects cyclic branches as soon as they arise rather than by worst-case exponential branching with respect to the size of φ. Moreover, by retaining the rooted-tree form from traditional tableaux, our calculus becomes amenable to the vast array of optimisation techniques which have proved essential for "practical" automated reasoning in very expressive description logics. Our calculus forms the basis for developing a uniform framework for the family of all fix-point logics of common knowledge. However, there is still no "free lunch" as, in the worst case, our method exhibits 2EXPTIME-behaviour. A prototype implementation can be found at twb.rsise.anu.edu.au which allows users to test formulae via a simple graphical interface.

1 Introduction and Motivation

The logic of common knowledge LCK is a multi-modal logic where for a finite number of agents $a_1, \ldots, a_n$, the expressions $[i]\varphi$ captures that "agent a_i knows φ"; $[E]\varphi$ captures that "every agent knows φ"; and $[C]\varphi$ captures that "φ is common knowledge". LCK is a fix-point logic since the equivalence $[C]\varphi \leftrightarrow [E](\varphi \wedge [C]\varphi)$ is valid, capturing the fact that the denotation of $[C]\varphi$ is the greatest fix-point of the recursive equation $\nu X.([E](\varphi \wedge X))$ [5]. The decision problem for LCK is known to be EXPTIME-complete [9] and automated reasoning in LCK is of fundamental importance in the area of intelligent agents.

* NICTA is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence program.

The traditional decision procedure for LCK is a two-pass tableau method derived from similar procedures for temporal logics [4, 8] which also have a fix-point nature. Given a formula φ, the first pass builds a cyclic graph where each node in the graph contains a subset of the Fisher-Ladner closure of φ. The second pass then prunes nodes of the graph that contain obvious contradictions like $\{p, \neg p\}$, and also prunes nodes that contain "eventualities" like $\langle C \rangle \psi$ which remain "unfulfilled" because no path from the node contains a node that contains ψ. The formula φ is unsatisfiable iff the root node is pruned. The disadvantage of such two-pass methods is that the first pass builds a graph which is *always* of exponential size relative to the size of the given formula φ. Thus, two-pass methods *always* exhibit the worst-case EXPTIME behaviour.

Because of its fix-point nature, it should be possible to extend the optimal resolution-based methods [3] and optimal automata-based [14] methods for fix-point temporal logics to LCK. But we know of no actual resolution methods or automata-based methods for LCK itself, although the resolution method of Dixon and Fisher [6] comes close since it includes a fix-point modality $[C]\varphi$, but does not seem to actually allow $[C]\varphi$ as an object level connective.

There is a also a decision procedure for LCK using a cut-free Gentzen-style sequent calculus [13]. Soundness, completeness and cut-elimination is proved by first using a Gentzen-style calculus containing an ω-rule with an infinite number of premises. Using the finite model property for LCK, the ω-rule is then "finitized" to have a finite number k of premises, where $k \approx O(2^{|\varphi|})$ and $|\varphi|$ is the size of the given formula φ. Although the resulting calculus is cut-free, the "finitized ω-rule" *always* has an exponential number of premises with respect to the size of the given formula φ, even when this is not necessary.

Thus, we know of no decision procedure for LCK which does not exhibit the worst-case exponential behaviour for *all* formulae.

Here we present a tableau-based decision procedure for LCK with the following properties: it is cut-free; the average-case behaviour is not necessarily exponential in the size of the given formula; the tableau is a finite rooted tree rather than a cyclic graph; the decision procedure requires only a single pass of the tableau which can be done as it is created; there is potential for parallelisation where each branch is farmed out to a different processor; and the method is amenable to the vast array of optimisations [11] which have led to "practical" tableau-based decision procedures for very expressive description logics [10].

Our tableaux require standard extra devices like "histories" to detect cyclic branches and slightly non-standard "variables" to pass information from children to parents in the tableau. But the only extra operations are those of set membership, set intersection, and the usual min/max operations on sets of integers.

The price we pay is that, in the worst-case, our tableaux require $O(2^{2^{|\varphi|}})$-time rather than $O(2^{|\varphi|})$-time relative to the size $|\varphi|$ of the given formula φ. We have implemented our procedure using the Tableau WorkBench (`twb.rsise.anu.edu.au`), a system for rapid prototyping of tableau/sequent calculi [1].

2 Syntax, Semantics and Hintikka Structures

Definition 1. *Let* AP *denote the set* $\{p_0, p_1, p_2, \dots\}$ *of propositional variables and* AG *the finite set* $\{0, 1, \dots, N_{\mathrm{AG}}\}$ *of* agents. *The set* Fml *of all formulae of the logic LCK is inductively defined as follows:*

1. $\mathrm{AP} \subset \mathrm{Fml}$;
2. *if* φ *and* ψ *are in* Fml, *then so are* $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $[C]\varphi$, *and* $\langle C\rangle\varphi$;
3. *if* φ *is in* Fml, *then so are* $[a]\varphi$ *and* $\langle a\rangle\varphi$ *for every* $a \in \mathrm{AG}$.

Additionally, we define $[E]\varphi := \bigwedge_{a\in\mathrm{AG}}[a]\varphi$ *and* $\langle E\rangle\varphi := \bigvee_{a\in\mathrm{AG}}\langle a\rangle\varphi$. *A formula of the form* $\langle a\rangle\varphi$, $[a]\varphi$, *or* $\langle C\rangle\varphi$ *is called an* $\langle-\rangle$-, $[-]$-, *or* $\langle C\rangle$*-formula, respectively. Let* $\mathrm{Fml}\langle\mathrm{C}\rangle$ *denote the set of all* $\langle C\rangle$*-formulae.*

Implication, equivalence, and $\top$ are not part of the core language but can be defined as $\varphi \to \psi := \neg\varphi \vee \psi$, $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$, and $\top := p_0 \vee \neg p_0$.

Definition 2. *A* transition frame *is a pair* (W, R) *where* W *is a non-empty set of worlds and* R *is a function that assigns to each agent* $a \in \mathrm{AG}$ *a binary relation* R_a *over* W. *We define* $R_{\mathrm{AG}} := \bigcup_{a\in\mathrm{AG}} R_a$.

Definition 3. *Let* (W, R) *be a transition frame. A* transition sequence σ *in* (W, R) *is a finite or infinite sequence of worlds in* W *with the following properties: if* σ *is an infinite sequence* $\sigma_0, \sigma_1, \sigma_2, \dots$, *then* $\sigma_i \, R_{\mathrm{AG}} \, \sigma_{i+1}$ *for all* $i \in \mathbb{N}$; *if* σ *is a finite sequence* $\sigma_0, \sigma_1, \dots, \sigma_n$, *then* $\sigma_i \, R_{\mathrm{AG}} \, \sigma_{i+1}$ *for all* $i < n$ *and there is no* $w \in W$ *such that* $\sigma_n \, R_{\mathrm{AG}} \, w$. *A* w-sequence σ *in* (W, R) *is a transition sequence in* (W, R) *with* $\sigma_0 = w$. *Let* $\mathcal{B}(w)$ *be the set of all* w*-sequences in* (W, R) *(we assume that* (W, R) *is clear from the context).*

Definition 4. *A* model $M = (W, R, L)$ *is a transition frame* (W, R) *and a labelling function* $L : W \to 2^{\mathrm{AP}}$ *which associates with each world* w *a set* $L(w)$ *of propositional variables true at world* w.

Definition 5. *Let* $M = (W, R, L)$ *be a model. The* satisfaction relation $\Vdash$ *is defined inductively as follows for each* $a \in \mathrm{AG}$*:*

$$
\begin{array}{ll}
M, w \Vdash p & \textit{iff } p \in L(s), \textit{ for } p \in \mathrm{AP} \\
M, w \Vdash \neg\psi & \textit{iff } M, w \nVdash \psi \\
M, w \Vdash \varphi \wedge \psi & \textit{iff } M, w \Vdash \varphi \;\&\; M, w \Vdash \psi \\
M, w \Vdash \varphi \vee \psi & \textit{iff } M, w \Vdash \varphi \textit{ or } M, w \Vdash \psi \\
M, w \Vdash [a]\varphi & \textit{iff } \forall v \in W.\, w \, R_a \, v \;\Rightarrow\; M, v \Vdash \varphi \\
M, w \Vdash \langle a\rangle\varphi & \textit{iff } \exists v \in W.\, w \, R_a \, v \;\&\; M, v \Vdash \varphi \\
M, w \Vdash [C]\varphi & \textit{iff } \forall \sigma \in \mathcal{B}(w).\, \forall i \in \mathbb{N}_{>0}.\, M, \sigma_i \Vdash \varphi \\
M, w \Vdash \langle C\rangle\varphi & \textit{iff } \exists \sigma \in \mathcal{B}(w).\, \exists i \in \mathbb{N}_{>0}.\, M, \sigma_i \Vdash \varphi \;\;.
\end{array}
$$

Traditionally, the semantics of $[C]\varphi$ is given with the help of iterations of the formula $[E]\varphi$ [7]. It is easy to show that both definitions are equivalent [2].

Table 1. Smullyan's $\alpha-$ and $\beta-$notation to classify formulae

α	α_1	α_2
$\varphi \wedge \psi$	φ	ψ
$[C]\varphi$	$[E]\varphi$	$[E][C]\varphi$

β	β_1	β_2
$\varphi \vee \psi$	φ	ψ
$\langle C\rangle\varphi$	$\langle E\rangle\varphi$	$\langle E\rangle\langle C\rangle\varphi$

Definition 6. *A formula* $\varphi \in \mathrm{Fml}$ *is* satisfiable *iff there is a model* $M = (W, R, L)$ *and some* $w \in W$ *such that* $M, w \Vdash \varphi$. *A formula* $\varphi \in \mathrm{Fml}$ *is* valid *iff* $\neg\varphi$ *is not satisfiable.*

Definition 7. *A formula* $\varphi \in \mathrm{Fml}$ *is in* negation normal form *if the symbol* $\neg$ *appears only immediately before propositional variables. For every formula* $\varphi \in \mathrm{Fml}$, *we can obtain a formula* $\mathrm{nnf}(\varphi)$ *in negation normal form by pushing negations inward as far as possible (*e.g. *by using de Morgan's laws) such that* $\varphi \leftrightarrow \mathrm{nnf}(\varphi)$ *is valid.*

Proposition 8. *In the notation of Table 1, the formulae of the form* $\alpha \leftrightarrow \alpha_1 \wedge \alpha_2$ *and* $\beta \leftrightarrow \beta_1 \vee \beta_2$ *are valid.*

Definition 9. *Let* $\varphi \in \mathrm{Fml}$ *be a formula in negation normal form. The* closure $\mathrm{cl}(\varphi)$ *of* φ *is the least set of formulae such that:*

1. *each subformula of* φ, *including* φ *itself, is in* $\mathrm{cl}(\varphi)$;
2. *if* $[C]\psi \in \mathrm{cl}(\varphi)$, *then* $[E]\psi \in \mathrm{cl}(\varphi)$ *and* $[E][C]\psi \in \mathrm{cl}(\varphi)$;
3. *if* $[E]\psi \in \mathrm{cl}(\varphi)$, *then* $[a]\psi \in \mathrm{cl}(\varphi)$ *for every* $a \in \mathrm{AG}$;
4. *if* $\langle C\rangle\psi \in \mathrm{cl}(\varphi)$, *then* $\langle E\rangle\psi \in \mathrm{cl}(\varphi)$ *and* $\langle E\rangle\langle C\rangle\psi \in \mathrm{cl}(\varphi)$;
5. *if* $\langle E\rangle\psi \in \mathrm{cl}(\varphi)$, *then* $\langle a\rangle\psi \in \mathrm{cl}(\varphi)$ *for every* $a \in \mathrm{AG}$.

Definition 10. *A* structure (W, R, L) [*for* $\varphi \in \mathrm{Fml}$] *is a transition frame* (W, R) *and a labelling function* $L : W \to 2^{\mathrm{Fml}}$ *which associates with each world* $w \in W$ *a set* $L(w)$ *of formulae* [*and has* $\varphi \in L(v)$ *for some world* $v \in W$].

Definition 11. *A* pre-Hintikka structure $H = (W, R, L)$ [*for* $\varphi \in \mathrm{Fml}$] *is a structure* [*for* φ] *that satisfies the following conditions for every* $w \in W$ *where* α *and* β *are formulae as defined in Table 1 and* $a \in \mathrm{AG}$:

$$\begin{aligned}
&\mathrm{H1} : \neg p \in L(w) \text{ and } p \in \mathrm{AP} \;\Rightarrow\; p \notin L(w) \;;\\
&\mathrm{H2} : \alpha \in L(w) \;\Rightarrow\; \alpha_1 \in L(w) \;\&\; \alpha_2 \in L(w) \;;\\
&\mathrm{H3} : \beta \in L(w) \;\Rightarrow\; \beta_1 \in L(w) \text{ or } \beta_2 \in L(w) \;;\\
&\mathrm{H4} : [a]\varphi \in L(w) \;\Rightarrow\; \forall v \in W.\, w\, R_a\, v \;\Rightarrow\; \varphi \in L(v) \;;\\
&\mathrm{H5} : \langle a\rangle\varphi \in L(w) \;\Rightarrow\; \exists v \in W.\, w\, R_a\, v \;\&\; \varphi \in L(v) \;;\\
&\mathrm{H6} : [E]\varphi \in L(w) \;\Rightarrow\; \forall a \in \mathrm{AG}.\, [a]\varphi \in L(w) \;;\\
&\mathrm{H7} : \langle E\rangle\varphi \in L(w) \;\Rightarrow\; \exists a \in \mathrm{AG}.\, \langle a\rangle\varphi \in L(w) \;.
\end{aligned}$$

A Hintikka structure $H = (W, R, L)$ [*for* $\varphi \in \mathrm{Fml}$] *is a pre-Hintikka structure* [*for* φ] *that additionally satisfies the following conditions:*

$$\mathrm{H8} : \langle C\rangle\varphi \in L(w) \;\Rightarrow\; \exists \sigma \in \mathcal{B}(w).\, \exists i \in \mathbb{N}_{>0}.\, \varphi \in L(\sigma_i) \;.$$

Proposition 12. *A formula* $\varphi \in \mathrm{Fml}$ *in negation normal form is satisfiable iff there exists a Hintikka structure for* φ*.*

3 A One-pass Tableau Algorithm for LCK

A *tableau algorithm* is a systematic search for a model of a formulae ϕ. Its data structures are single-rooted finite trees – called *tableaux* – where each node is labelled with a set of formulae that is derived from the formula set of its parent according to some given rules (unless it is the root, of course). The algorithm starts with a single node that is labelled with the singleton set $\{\phi\}$ and incrementally expands the tableau by applying the rules mentioned before to its leaves. The result of the tableau algorithm is a tableau where no more rules can be applied. Such tableaux are called *expanded*. On any branch of the tableau, a node t is an ancestor of a node s iff t lies above s on the unique path from the root down to s.

An expanded tableau can be associated with a pre-Hintikka structure H for ϕ, and ϕ is satisfiable if and only if H is a Hintikka structure for ϕ. To be able to determine whether H is a Hintikka structure, the algorithm stores additional information with each node of the tableau using *histories* and *variables*. A history is a mechanism for collecting extra information during proof search and passing it from parents to children. A variable is a mechanism to propagate information from children to parents.

In the following, we restrict ourselves to the tableau algorithm for LCK.

Definition 13. *A* tableau node x *is of the form* $(\Gamma :: \mathrm{HAg}, \mathrm{HCr} :: \mathrm{mrk}, \mathrm{uev})$ *where:*

Γ *is a set of formulae;*
HAg *is a partial function from formulae to agents;*
HCr *is a list of the formula sets of some designated ancestors of* x*;*
mrk *is a boolean valued variable indicating whether the node is marked; and*
uev *is a partial function from formulae to* $\mathbb{N}_{>0}$*.*

The list HCr and the partial function HAg are histories, *i.e.* their values in a node are determined by the parent node, whereas mrk and uev are variables, *i.e.* their values in a node are determined by the children. In the following we call tableau nodes just nodes when the meaning is clear.

We postpone the definition of a rule for a moment and proceed with the definition of a tableau.

Definition 14. *A* tableau *for a set* $\Gamma \subseteq \mathrm{Fml}$ *and a list of formula sets* HCr *is a tree of tableau nodes with root* $(\Gamma :: \mathrm{HAg}, \mathrm{HCr} :: \mathrm{mrk}, \mathrm{uev})$ *where the children of a node* x *are obtained by a single application of a rule to* x *(*i.e. *only one rule can be applied to a node). A tableau is* expanded *if no rules can be applied to any of its leaves.*

Note that mrk and uev in the definition are not given but are part of the result as they are determined by the children of the root.

Definition 15. *The partial function* $\mathrm{uev}_\perp : \mathrm{Fml} \rightharpoonup \mathbb{N}_{>0}$ *is the constant function that is undefined for all formulae (*i.e. $\mathrm{uev}_\perp(\psi) = \perp$ *for all* $\psi \in \mathrm{Fml}$*). Analogously, the partial function* $\mathrm{HAg}_\perp : \mathrm{Fml} \rightharpoonup \mathrm{AG}$ *is undefined everywhere.*

Note 16. In the following, we use Λ to denote a set containing only propositional variables or their negations (*i.e.* $\varphi \in \Lambda \Rightarrow \exists p \in \mathrm{AP}.\, \varphi = p$ or $\varphi = \neg p$). To focus on the "important" parts of the rule, we use "$\cdots$" for the "unimportant" parts which are passed from node to node unchanged (*e.g.* $(\Gamma :: \cdots :: \cdots)$).

3.1 The Rules

Terminal Rule.

$$(id)\ \ \frac{(\Gamma :: \cdots :: \mathrm{mrk}, \mathrm{uev})}{}\ \{p, \neg p\} \subseteq \Gamma \text{ for some } p \in \mathrm{AP}$$

with $\mathrm{mrk} := \mathbf{true}$ and $\mathrm{uev} := \mathrm{uev}_\perp$. The intuition is that the node is "closed" so we pass this information up to the parent by putting mrk to **true**, and putting uev as undefined for all formulae.

Linear (α) Rules.

$$(\wedge)\ \ \frac{(\varphi \wedge \psi\ ;\ \Gamma :: \cdots :: \cdots)}{(\varphi\ ;\ \psi\ ;\ \Gamma :: \cdots :: \cdots)}$$

$$([E])\ \ \frac{([E]\varphi\ ;\ \Gamma :: \cdots :: \cdots)}{([1]\varphi\ ;\ [2]\varphi\ ;\ \ldots\ ;\ [N_{\mathrm{AG}}]\varphi\ ;\ \Gamma :: \cdots :: \cdots)}$$

$$([C])\ \ \frac{([C]\varphi\ ;\ \Gamma :: \cdots :: \cdots)}{([E]\varphi\ ;\ [E][C]\varphi\ ;\ \Gamma :: \cdots :: \cdots)}$$

The $\wedge$-rule is standard and the $[E]$-rule just unfolds the definition of $[E]\varphi$. The $[C]$-rule capture the fix-point nature of the corresponding formulae according to Prop. 8. These rules do not modify the histories or variables at all.

Universal Branching (β) Rules.

$$(\vee)\ \ \frac{(\varphi_1 \vee \varphi_2\ ;\ \Gamma :: \cdots :: \mathrm{mrk}, \mathrm{uev})}{(\varphi_1\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_1, \mathrm{uev}_1)\ |\ (\varphi_2\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_2, \mathrm{uev}_2)}$$

$$(\langle E\rangle)\ \ \frac{(\langle E\rangle\varphi\ ;\ \Gamma :: \mathrm{HAg}, \cdots :: \mathrm{mrk}, \mathrm{uev})}{\begin{array}{l}\langle 1\rangle\varphi\ ;\ \Gamma \\ :: \mathrm{HAg}_1, \cdots :: \mathrm{mrk}_1, \mathrm{uev}_1\end{array}\ \Big|\ \cdots\ \Big|\ \begin{array}{l}\langle N_{\mathrm{AG}}\rangle\varphi\ ;\ \Gamma \\ :: \mathrm{HAg}_{N_{\mathrm{AG}}}, \cdots :: \mathrm{mrk}_{N_{\mathrm{AG}}}, \mathrm{uev}_{N_{\mathrm{AG}}}\end{array}}$$

$$(\langle C\rangle)\ \ \frac{(\langle C\rangle\varphi\ ;\ \Gamma :: \cdots :: \mathrm{mrk}, \mathrm{uev})}{(\langle E\rangle\varphi\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_1, \mathrm{uev}_1)\ |\ (\langle E\rangle\langle C\rangle\varphi\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_2, \mathrm{uev}_2)}$$

with:

$$\mathrm{HAg}_i(\psi) := \begin{cases} i & \text{if } \psi \in \mathrm{Fml}\langle \mathrm{C}\rangle \ \&\ \psi = \varphi \\ \mathrm{HAg}(\psi) & \text{otherwise} \end{cases}$$
$$\text{for } i = 1, \ldots, N_{\mathrm{AG}} \text{ in the } \langle E\rangle\text{-rule only;}$$

$$n := \begin{cases} N_{\mathrm{AG}} & \text{for the } \langle E\rangle\text{-rule} \\ 2 & \text{otherwise} \end{cases}$$

$$\mathrm{mrk} := \mathrm{mrk}_1 \ \&\ \ldots \ \&\ \mathrm{mrk}_n$$

$$\mathrm{excl}_\phi(f)(\psi) := \begin{cases} \bot & \text{if } \psi = \phi \\ f(\psi) & \text{otherwise} \end{cases}$$

$$\mathrm{uev}'_1 := \begin{cases} \mathrm{excl}_{\langle C\rangle\varphi}(\mathrm{uev}_1) & \text{for the } \langle C\rangle\text{-rule} \\ \mathrm{uev}_1 & \text{otherwise} \end{cases}$$

$$\mathrm{uev}'_i := \mathrm{uev}_i \text{ for } i = 2, \ldots, n$$

$$\min_\bot\{f_1, \ldots, f_k\}(\psi) := \begin{cases} l & \text{if } k > 0 \ \&\ \forall i \in \{1, \ldots, k\}.\ f_i(\psi) \neq \bot \ \& \\ & \quad l = \min\{f_1(\psi), \ldots, f_k(\psi)\} \\ \bot & \text{otherwise} \end{cases}$$

$$\mathrm{uev} := \min_\bot\{\mathrm{uev}'_i \mid i \in \{1, \ldots, n\} \ \&\ mrk_i = \mathbf{false}\}$$

The $\vee$-rule is standard and the $\langle E\rangle$-rule just unfolds the definition of $\langle E\rangle\varphi$. The $\langle C\rangle$-rule captures the fix-point nature of the $\langle C\rangle$-formulae, according to Prop. 8. The intuitions of the definitions of the histories and variables are:

HAg_i: in the $\langle E\rangle$-rule, if the principal formula $\langle E\rangle\varphi$ is $\langle E\rangle\langle C\rangle\chi$ (*i.e.* we have $\varphi \in \mathrm{Fml}\langle \mathrm{C}\rangle$), the value of φ in HAg_i is set to the corresponding agent a_i of the child to indicate that this child is "tracking" $\langle E\rangle\langle C\rangle\chi$;

mrk: the value of the variable mrk is **true** if the node is "closed", so the definition of mrk just captures the "universal" nature of these rules whereby the parent node is closed if all children are closed;

excl: the definition of $\mathrm{excl}_\phi(f)(\psi)$ just ensures that $\mathrm{excl}_\phi(f)(\phi)$ is undefined;

uev'_1: the definition of uev'_1 ensures that its value is undefined for the principal formulae of the $\langle C\rangle$-rule since the first child is charged with fulfilling $\langle C\rangle\varphi$; the other uev'_i for $i > 1$ are just there to avoid a case distinction in the definition of uev;

$\min_\bot$: the definition of $\min_\bot$ ensures that we take the minimum of all $f_i(\psi)$ only when all functions are defined for ψ;

uev: we only consider the uev'_i of unmarked (*i.e.* unclosed) children. If all children are "closed", then the argument to $\min_\bot$ is the empty set with $k = 0$, so uev is undefined everywhere. Otherwise, for a given formula ψ, we define $\mathrm{uev}(\psi)$ by first ensuring that $\mathrm{uev}'(\psi)$ is defined for *every* unclosed child, which means that none of the children "fulfil" ψ, and then assigning $\mathrm{uev}(\psi)$ to be their minimum value, which ensures that we keep the "highest" looping non-fulfilling path. In particular, if the first child is unmarked in the $\langle C\rangle$-rule, then the first child fulfils $\langle C\rangle\varphi$, so $\mathrm{uev}(\langle C\rangle\varphi)$ is undefined for the principal formula $\langle C\rangle\varphi$ via the definition of uev'_i.

Existential Branching Rule.

$$(\langle a\rangle)\ \frac{\begin{array}{l}\langle a_1\rangle\varphi_1;\ldots;\langle a_n\rangle\varphi_n\ ;\ \langle a_{n+1}\rangle\varphi_{n+1};\ldots;\langle a_{n+m}\rangle\varphi_{n+m}\ ;\\ [1]\Delta_1;\ldots;[N_{\mathrm{AG}}]\Delta_{N_{\mathrm{AG}}}\ ;\ \Lambda\\ ::\mathrm{HAg},\mathrm{HCr}::\mathrm{mrk},\mathrm{uev}\end{array}}{\begin{array}{l}\varphi_1\ ;\ \Delta_{a_1}\\ ::\mathrm{HAg}_\perp,\mathrm{HCr}_1::\mathrm{mrk}_1,\mathrm{uev}_1\end{array}\ \big|\ \cdots\ \big|\ \begin{array}{l}\varphi_n\ ;\ \Delta_{a_n}\\ ::\mathrm{HAg}_\perp,\mathrm{HCr}_n::\mathrm{mrk}_n,\mathrm{uev}_n\end{array}}$$

where:

(1) $\{p,\neg p\}\not\subseteq\Lambda$
(2) $n+m\geq 0$
(3) $\forall i\in\{1,\ldots,n+m\}.\ a_i\in\mathrm{AG}$
(4) $\forall i\in\{1,\ldots,n\}.\ \forall j\in\{1,\ldots,\mathrm{len}(\mathrm{HCr})\}.\ \{\varphi_i\}\cup\Delta_{a_i}\neq\mathrm{HCr}[j]$
(5) $\forall k\in\{n+1,\ldots,n+m\}.\ \exists j\in\{1,\ldots,\mathrm{len}(\mathrm{HCr})\}.\ \{\varphi_k\}\cup\Delta_{a_k}=\mathrm{HCr}[j]$

with:

$$\mathrm{HCr}_i := \mathrm{HCr}\ @\ [\{\varphi_i\}\cup\Delta_{a_i}]\ \text{for}\ i=1,\ldots,n$$

$$\mathrm{mrk} := \textstyle\bigvee_{i=1}^{n}\mathrm{mrk}_i\ \text{or}\ \exists i\in\{1,\ldots,n\}.\ \exists\psi\in\{\varphi_i\}\cup\Delta_{a_i}.\ \perp\neq\mathrm{uev}_i(\psi)>\mathrm{len}(\mathrm{HCr})$$

$$\mathrm{uev}_k(\cdot) := j\in\{1,\ldots,len(\mathrm{HCr})\}\ \text{such that}\ \{\varphi_k\}\cup\Delta_{a_k}=\mathrm{HCr}[j]\ \text{for}\ k=n+1,\ldots,n+m$$

$$\mathrm{uev}(\psi) := \begin{cases}\mathrm{uev}_i(\psi) & \text{if}\ \psi\in\mathrm{Fml}\langle\mathrm{C}\rangle\ \&\ \psi=\varphi_i\ \&\ \mathrm{HAg}(\psi)=a_i\\ & \quad\text{for an}\ i\in\{1,\ldots,n+m\}\\ \perp & \text{otherwise}\end{cases}$$

Some intuitions are in order:

(1) The $\langle a\rangle$-rule is applicable if the parent node contains no α- or β-formulae and Λ, which contains atoms and their negations only, contains no atomic contradictions.
(2) Both n and m can be zero.
(4) If $n>0$ then each $\langle a_i\rangle\varphi_i$ for $1\leq i\leq n$ is not "blocked" by an ancestor, and has a child containing $\varphi_i;\Delta_{a_i}$, thereby generating the required $\langle-\rangle$-successor.
(5) If $m>0$ then each $\langle a_k\rangle\varphi_k$ for $n+1\leq k\leq m$ is "blocked" from creating its required child $\varphi_k;\Delta_{a_k}$ because some ancestor does the job.

HAg_i: we reset the partial functions HAg_i of the children so that they are undefined for all formulae since we no longer need to "track" them.

HCr_i: is just the HCr of the parent but with an extra entry to extend the "history" of nodes on the path from the root down to the i^{th} child.

mrk: captures the "existential" nature of this rule whereby the parent is marked if some child is closed or if some child contains a formula whose uev is defined and "loops" lower than the parent. Moreover, if n is zero, then mrk is set to **false** to indicate that this branch is "open" (*i.e.* not "closed").

uev_k: for $n+1 \leq k \leq n+m$ the k^{th} child is blocked by a proxy child higher in the branch. For every such k we set uev_k to be the *constant* function which maps every formula to the level of this proxy child. Note that this is just a temporary function used to define uev as explained next.

$\text{uev}(\psi)$: for a $\langle C\rangle$-formula $\psi = \langle C\rangle\chi$, we check whether there is a principal formula $\langle a_i\rangle\varphi_i$ so that $\varphi_i = \psi$ and $a_i = \text{HAg}(\psi)$ indicating that ψ is being tracked by the i^{th} child and hence this i^{th} child has the responsibility to fulfil $\langle C\rangle\chi$. If no such formulae exist, we put uev to be undefined. Otherwise, we take uev of ψ from the corresponding child if $\langle a_i\rangle\varphi_i$ is "unblocked", or set it to be the *level* of the proxy child higher in the branch if it is "blocked". For all other formulae, put uev to be undefined. The intuition is that a defined $\text{uev}(\psi)$ tells us that there is a "loop" which starts at the parent and eventually "loops" up to some blocking node higher up on the current branch. The actual value of $\text{uev}(\psi)$ tells us the level of the proxy because we cannot distinguish whether this "loop" is "good" or "bad" until we backtrack up to that level.

Note that the $\langle a\rangle$-rule and the *id*-rule are mutually exclusive since their side-conditions cannot be simultaneously true.

3.2 Fullpaths, Virtual Successors and Termination of Proof Search

Definition 17. *Let $G = (W, R)$ be a directed graph (*e.g. *a tableau where R is just the child-of relation between nodes). A* [full]path π *in G is a finite* [*infinite*] *sequence $x_0, x_1, x_2, \ldots$ of nodes in W such that $x_i\, R\, x_{i+1}$ for all x_i except the last node if π is finite. An x-*[full]path π *in G is a* [*full*]*path in G that has $x_0 = x$.*

Definition 18. *Let $x = (\Gamma :: \text{HAg}, \text{HCr} :: \text{mrk}, \text{uev})$ be a tableau node, φ a formulae, and Δ a set of formulae. We write $\varphi \in x$ [$\Delta \subseteq x$] to denote $\varphi \in \Gamma$ [$\Delta \subseteq \Gamma$]. The elements* HAg, HCr, mrk, *and* uev *of x are denoted by HAg_x, HCr_x, mrk_x, and uev_x, respectively. The node x is* marked *iff mrk_x is set to* **true**.

Definition 19. *Let x be an $\langle a\rangle$-node in a tableau T (*i.e. *an $\langle a\rangle$-rule was applied to x). Then x is also called a* **state** *and the children of x are called* **pre-states**. *Using the notation of the $\langle a\rangle$-rule, a $\langle -\rangle$-formula $\langle a_i\rangle\varphi_i \in x$ is* **blocked** *iff $n + 1 \leq i \leq n + m$. For every blocked $\langle a_i\rangle\varphi_i \in x$ there exists a unique pre-state y on the path from the root of T to x such that $\{\varphi_i\} \cup \Delta_{a_i}$ equals the set of formulae of y. We call y the* **virtual successor** *of $\langle a_i\rangle\varphi_i$. For every not blocked $\langle a_i\rangle\varphi_i$ of x, the* **successor** *of $\langle a_i\rangle\varphi_i$ is the i^{th} child of the $\langle a\rangle$-rule.*

Note that a state is just another term for an $\langle a\rangle$-node, whereas a pre-state can be any type of node (it may even be a state).

Proposition 20 (Termination). *Let $\phi \in$ Fml be a formula in negation normal form. Any tableau T for a node $(\{\phi\} :: \cdots :: \cdots)$ is a finite tree, hence the procedure that builds a tableau always terminates.*

Proof. It is obvious that T is a tree, that every node in T can contain only formulae of the closure $\mathrm{cl}(\phi)$, and that $\mathrm{cl}(\phi)$ is finite. Hence, there are only a finite number of different sets that can be assigned to the nodes, in particular the pre-states. As the $\langle a \rangle$-rule guarantees that all pre-states on a path possess different sets of formulae, there can only be a finite number of pre-states on a path. Furthermore, from any pre-state, there are only a finite number of consecutive nodes on a path until we reach a state. As every state in a path is followed by a pre-state and there are only a finite number of pre-states, all paths in T must be finite. This, the obvious fact that every node in T has finite degree, and König's lemma complete the proof. □

3.3 Soundness

Let $\phi \in \mathrm{Fml}$ be a formula in negation normal form and T an expanded tableau with root $r = (\{\phi\} :: \mathrm{HAg}_{\perp}, [] :: \mathrm{mrk}, \mathrm{uev})$: that is, the initial formula set is $\{\phi\}$, the initial HAg is undefined everywhere, and the initial HCr is the empty list.

Theorem 21. *If r is not marked then there exists a Hintikka structure for ϕ.*

Proof. By construction, T is a finite tree. Let T_{p} ("p" for pruned) be the subgraph that consists of all nodes x having the following property: there is a path of unmarked nodes from r to x inclusive. The edges of T_{p} are exactly the edges of T that connect two nodes in T_{p}. Clearly, T_{p} is also a finite tree with root r. Intuitively, T_{p} is the result of pruning all subtrees of T that have a marked root.

As *id*-nodes are marked by construction, all leaves of T_{p} must be states where all $\langle - \rangle$-formulae (if any) are blocked. Hence every $\langle - \rangle$-formula $\langle a \rangle \varphi$ of every leaf x has a virtual successor, which lies on the path from r to x. We extend T_{p} to a finite cyclic tree T_{l} ("l" for looping) by doing the following for every leaf x: for every $\langle - \rangle$-formula $\langle a \rangle \varphi \in x$, we add the edge (x, y) where y is the virtual successor of $\langle a \rangle \varphi$. These new edges are called *backward edges*.

Following Ben-Ari [4], the finite cyclic tree T_{l} is then "unwound" to a finite cyclic tree T_{u} as described next. The nodes of T_{u} are instances of nodes of T_{l}. We denote the nodes of T_{u} by $x', x'', \ldots$ when they are instances of $x \in T_{\mathrm{l}}$. Let r' be the root of T_{u}. We define T_{u} inductively by applying the following procedure to every node in T_{u} exactly once (starting with the only possibility r').
Notation: Following Ben-Ari [4], we call a node $x \in T_{\mathrm{l}}$ an *alternative node* iff it is a β-node that still has more than one unmarked child. To keep things simple, we assume in the rest of this proof that we have only two agents. This allows us to treat all β-rules equally. It should be obvious that the following procedure can easily be extended to $\langle E \rangle$-nodes with higher degrees (*i.e.* with $N_{\mathrm{AG}} > 2$ agents). If π is a path and x is a node that appears exactly once in π, then $\pi(x)$ denotes the suffix path of π starting at x.

Let x' be a node in T_{u} that has not been processed yet and π be *the* path from r' down to x' (including r' but *excluding* x'). If we stipulate that every node in T_{u} appears at most once in π, then π is unique. If there is an x'' on π (*i.e.* x'' is another instance of x) such that every alternative node $y \in T_{\mathrm{l}}$ which has an

instance y' on $\pi(x'')$ has at least *two* instances on $\pi(x'')$, then we identify x' with x'' by removing x' and redirecting all edges incident on x' to x''. Otherwise we distinguish whether or not $x \in T_{\mathrm{l}}$ is an alternative node.

If $x \in T_{\mathrm{l}}$ is not an alternative node, then for every child $x_i \in T_{\mathrm{l}}$ of x, create the i^{th} child of x' in T_{u} as the new node instance x'_i. If $x \in T_{\mathrm{l}}$ is an alternative node, let $x_1 \in T_{\mathrm{l}}$ and $x_2 \in T_{\mathrm{l}}$ be the children of x and let k be the number of instances of x on π (remember that π excludes x'). If $k = 0$ then create the new node x'_1 as the (only) child of x' in T_{u} (the choice to insert an instance of x_1 is arbitrary). If $k > 0$ then let x'' be the $k-1$'st instance of x on π. If x''_1 (x''_2) is the child of x'' then create the new node x'_2 (x'_1) as the (only) child of x' in T_{u}: that is, we alternate choices [4].

Since there are only a finite number of alternative nodes in T_{l}, it is not too hard to see that we only insert a finite number of nodes in T_{u}. In particular, all β-nodes in T_{u} have exactly one child, and every fullpath π_{u} in T_{u} canonically yields a fullpath π_{l} in T_{l} with the following property: if an alternative node $x \in T_{\mathrm{l}}$ occurs infinitely often on π_{l}, then both children of x occur infinitely often in π_{l} too. We call π_{l} the *projection* of π_{u}.

Finally, following Ben-Ari [4], the cyclic tree T_{u} is used to generate a structure $H = (W, R, L)$ as described next. Let W be the set of all states of T_{u}. For every $a \in \mathrm{AG}$ and every $s, t \in W$, let $s\, R_a\, t$ iff s contains a $\langle - \rangle$-formula $\langle a \rangle \psi$ and there exists a path $x_0 = s, x_1, \ldots, x_{k+1} = t$ in T_{u} such that x_1 is the (possibly virtual) successor of $\langle a \rangle \psi$ and each $x_i, 1 \leq i \leq k$ is an α- or a β-node. Note that $s\, R_a\, t$ and $s\, R_b\, t$ is possible for $a \neq b$, because a pre-state might be the virtual successor of more than one $\langle - \rangle$-formula in s.

If we consider the root of T_{u} as a pre-state for a moment, it is not hard to see that for every state s of T_{u} there exists a unique pre-state x such that there is a (unique) path τ of the form $x_0 = x, x_1, \ldots, x_{k+1} = s$ in T_{u} where each $x_i, 0 \leq i \leq k$ is an α- or a β-node. We set $L(s)$ to be the union of all formulae of all nodes on τ. Intuitively, we form $L(s)$ by adding back all the principal formulae of the α- and β-rules which were applied to x to obtain s.

It is not too hard to check that H is a pre-Hintikka structure for ϕ. Moreover, it is an easy consequence from Lemma 22 that H is even a Hintikka structure for ϕ. This concludes our proof. □

Lemma 22. *Let $\langle C \rangle \varphi \in y \in T_{\mathrm{u}}$. Then there exists a y-fullpath π_0 in T_{u} with the following property: there exists a node $x_0 \in \pi_0$ such that $\varphi \in x_0$ and there exists a state $s_0 \in \pi_0$ which appears strictly before x_0 in π_0.*

Proof. We inductively construct an y-(full)path π_0. Note that y is not a state as it contains $\langle C \rangle \varphi$. Also remember that states are the only nodes in T_{u} that offer us a choice.

Step 1: We start at y and follow the (unique) path in T_{u} until we reach the first state s. At some point in between, we pass the β-node z that deals with $\langle C \rangle \varphi$. According to the construction of T_{u}, the (only) child of z is either a left β-child z_1 where $\langle C \rangle \varphi$ is deconstructed into $\langle E \rangle \varphi$, or a right β-child z_2 where $\langle C \rangle \varphi$ is built-up into $\langle E \rangle \langle C \rangle \varphi$. In the first case, since z_1 contains $\langle E \rangle \varphi$, it is easy to see that s

must contain a $\langle - \rangle$-formula $\langle a \rangle \varphi$ for some $a \in \mathrm{AG}$. Hence, we can choose the (possibly virtual) successor x_s of $\langle a \rangle \varphi$ as the successor of s in π_0. As x_s contains φ by construction, let $x_0 := x_s$ and $s_0 := s$ and we are done. In the second case where z_2 contains $\langle E \rangle \langle C \rangle \varphi$, we choose the successor of s in π_0 as described next.

Since $\langle E \rangle \langle C \rangle \varphi \in z_2$, we must pass at least one β-node dealing with $\langle E \rangle \langle C \rangle \varphi$ between z and s on the path π_0. The last such node guarantees that there is a $\langle - \rangle$-formula $\langle a_j \rangle \varphi_j$ in s such that $\langle C \rangle \varphi = \varphi_j$ and $\mathrm{HAg}_s(\langle C \rangle \varphi) = a_j$ by the definition of the $\langle E \rangle$-rule. We take the (possibly virtual) successor x_s of $\langle a_j \rangle \varphi_j$ as the successor of s on π_0. This means that $\mathrm{uev}_s(\langle C \rangle \varphi)$ is either the "level" of x_s if x_s is a virtual successor, or we have $\mathrm{uev}_s(\langle C \rangle \varphi) = \mathrm{uev}_{x_s}(\langle C \rangle \varphi)$ otherwise.

As x_s contains $\langle C \rangle \varphi$ by construction, we repeat Step 1 for x_s and keep on repeating Step 1 as long as we end up at a node containing $\langle C \rangle \varphi$. There are two possibilities: either we stop eventually because we find a left β-child that deconstructed $\langle C \rangle \varphi$ into $\langle E \rangle \varphi$, or we get a fullpath π_0 with the following properties: (†) every pre-state on π_0 contains $\langle C \rangle \varphi$ and every state s on π_0 has a pre-state x_s as successor on π_0 such that $\mathrm{uev}_s(\langle C \rangle \varphi)$ is either the "level" of x_s if x_s is a virtual successor (of the corresponding $\langle - \rangle$-formula in s), or we have $\mathrm{uev}_s(\langle C \rangle \varphi) = \mathrm{uev}_{x_s}(\langle C \rangle \varphi)$ otherwise. In the first possibility, we set x_0 and s_0 as described above and we are done; in the second possibility, let s_0 be the first state that appears on π_0 (*i.e.* there is no other state appearing strictly before s_0 on π_0). There are two further possibilities: if some node on π_0 that appears after s_0 contains φ, we set x_0 to be that node and are obviously done; otherwise we derive a contradiction as shown next.

Let the fullpath π be the projection of π_0 in T_l where the first nodes up to the first occurrence of s_0 inclusive have been removed. By assumption, none of the nodes on π contains φ. Furthermore, π inherits the properties (†) of π_0 that we have stated above.

For a node $x \in T_\mathrm{l}$, let π^r_x denote the unique path from the root r to x in T_l that does not use backward edges (*i.e.* π^r_x is also a path in the original tree T).

It is not too hard to see that HCr_x contains the formula sets of all pre-states on π^r_x in the correct order. As the formula sets are different for different pre-states in π^r_x by construction of the tableau, we can essentially view HCr_x as a list of pre-states on π^r_x.

Due to the construction of T_l, the fullpath π must use infinitely many backward edges. Hence, at least one of the finitely many pre-states in T_l must appear infinitely often on π. Let $x_\mathrm{h} \in T_\mathrm{l}$ be such a pre-state but such that no pre-state on $\pi^r_{x_\mathrm{h}}$ that is different from x_h appears infinitely often on π. Intuitively, x_h is the "highest" state that appears infinitely often, and its existence is guaranteed by the fact that π^r_x is finite for every $x \in T_\mathrm{l}$ by construction. Let $h_0 := \mathrm{len}(\mathrm{HCr}_{x_\mathrm{h}})$ denote the length of $\mathrm{HCr}_{x_\mathrm{h}}$. From what we have said above, it follows that h_0 equals the number of pre-states on $\pi^r_{x_\mathrm{h}}$.

We now consider the subtree T_0 of T_p rooted at x_h (*i.e.* we ignore the backward edges) and show the following claim:

Claim: for every node $x \in T_0$ that appears infinitely often on π, there exists a number $k_x \geq h_0$ such that $\mathrm{uev}_x(\langle C \rangle \varphi) = k_x$.

In particular, this yields $\mathrm{uev}_{x_\mathrm{h}}(\langle C\rangle\varphi) \geq h_0$; but this and the fact that $h_0 = \mathrm{len}(\mathrm{HCr}_{x_\mathrm{h}})$ means that the parent s of x_h, which is a state with $\mathrm{len}(\mathrm{HCr}_s) = h_0 - 1$, would have been marked by the second clause in the evaluation of mrk in the $\langle a\rangle$-rule. Hence, we have derived a contradiction to the fact that x_h is in T_l. The proof of the claim proceeds by structural induction on T_0 as shown next.

Proof of claim: Let x be a leaf of T_0 that appears infinitely often on π. Then x is a state where all $\langle -\rangle$-formulae are blocked and it contains at least one $\langle -\rangle$-formula. Furthermore, let x_1 be an immediate successor of x on π that appears infinitely often, too. To be able to use the notation of the $\langle a\rangle$-rule, we assume that x_1 is the virtual successor of the $\langle -\rangle$-formula $\langle a_k\rangle\varphi_k \in x$ such that $\langle C\rangle\varphi = \varphi_k$, $\mathrm{HAg}_x(\langle C\rangle\varphi) = a_k$, and hence, $\mathrm{uev}_x(\langle C\rangle\varphi) = \mathrm{uev}_k(\cdot)$. The existence of $\langle a_k\rangle\varphi_k$ is guaranteed by the construction of π_0 – and hence π – at the beginning of this proof.

By construction both x_h and x_1 are on π^r_x. Because of the properties of x_h, the node x_1 is either equal to x_h or appears after x_h on π^r_x. Remembering our connection between HCr_x and π^r_x, the application of the $\langle a\rangle$-rule on x gives us the constant function $\mathrm{uev}_k(\cdot) = h' \geq h_0$; but this implies $\mathrm{uev}_x(\langle C\rangle\varphi) = \mathrm{uev}_k(\cdot) \geq h_0$, which is exactly our claim.

If x is an α-node in T_0 that appears infinitely often on π, its child appears infinitely often on π also. As uev_x is obtained from the child unmodified, the claim follows directly from the induction hypothesis.

Let x be a β-node in T_0 that appears infinitely often on π. We know from the construction of T_p that every child of x which is not marked appears infinitely often in π since π is the projection of π_0 in T_u. Hence we can apply the induction hypothesis to all unmarked children of x. If x does not deconstruct $\langle C\rangle\varphi$, our claim follows by looking at the β-rule. If x does deconstruct $\langle C\rangle\varphi$, we additionally have to use the fact that the first child of x in T must be marked (*i.e.* it is not in T_l), because we would have stopped constructing π_0 in the first part of the proof otherwise. This is needed to ensure that $\langle C\rangle\varphi$ is defined in uev_x.

If x is an $\langle a\rangle$-node (*i.e.* a state) that appears infinitely often on π, it must have a child x_1 in T_l that appears infinitely often on π, too. To be able to use the notation of the $\langle a\rangle$-rule, we assume that x_1 is the (possibly virtual) successor of the $\langle -\rangle$-formula $\langle a_k\rangle\varphi_k \in x$ such that $\langle C\rangle\varphi = \varphi_k$, $\mathrm{HAg}_x(\langle C\rangle\varphi) = a_k$, and hence $\mathrm{uev}_x(\langle C\rangle\varphi) = \mathrm{uev}_k(\langle C\rangle\varphi)$. The existence of $\langle a_k\rangle\varphi_k$ is guaranteed by the construction of π_0 – and hence π – at the beginning of this proof.

The edge between x and x_1 can either be a forward edge (*i.e.* an edge that is already present in T) or a backward edge. In the first case, x_1 is also in T_0. Hence, we can apply the induction hypothesis for x_1 to obtain $\mathrm{uev}_{x_1}(\langle C\rangle\varphi) \geq h_0$. As $\mathrm{uev}_{x_1} = \mathrm{uev}_k$ in our notation of the $\langle a\rangle$-rule, this implies $\mathrm{uev}_x(\langle C\rangle\varphi) = \mathrm{uev}_{x_1}(\langle C\rangle\varphi) \geq h_0$. In the second case we have already proven the claim in the "leaf case". This concludes the proof. □

3.4 Completeness

Let $\phi \in \mathrm{Fml}$ be a formula in negation normal form and T an expanded tableau with root $r = (\{\phi\} :: \mathrm{HAg}_{\perp}, [] :: \mathrm{mrk}, \mathrm{uev})$: that is, the initial formula set is $\{\phi\}$, the initial HAg is undefined everywhere, and the initial HCr is the empty list.

Theorem 23. *For every marked node $x = (\Gamma :: \cdots :: \cdots)$ in T, the formula $\bigwedge_{\varphi \in \Gamma} \varphi$ is not satisfiable. In particular, if r is marked, then ϕ is not satisfiable.*

Proof. We use structural induction on T.

If a leaf $x \in T$ is marked, it must be an *id*-node as a state with no children is always unmarked. Hence, our theorem follows from the fact that $\{p, \neg p\} \subseteq x$ for some $p \in \mathrm{AP}$.

If x is a marked α-node, then its child is marked as well so we can apply the induction hypothesis and the claim follows from the definition of $[E]$ or the fact that – in the sense of Table 1 – the formulae of the form $\alpha \leftrightarrow \alpha_1 \wedge \alpha_2$ are valid (Prop. 8).

If x is a marked β-node, then both children are marked as well so we can apply the induction hypothesis and the claim follows from the definition of $\langle E \rangle$ or the fact that – in the sense of Table 1 – the formulae of the form $\beta \leftrightarrow \beta_1 \vee \beta_2$ are valid (Prop. 8).

If x is a marked $\langle a \rangle$-node (*i.e.* a marked state), then it has at least one child and there are two possibilities, for why it has been marked by the $\langle a \rangle$-rule:

(1) Some child x_0 of x is marked;
(2) Some unmarked child x_0 of x contains a $\langle C \rangle \varphi$ such that $\mathrm{uev}_{x_0}(\langle C \rangle \varphi) > h := \mathrm{len}(\mathrm{HCr}_x)$;

In the rest of the proof, let Γ_y denote the set of formulae of the node y. We say that a finite set of formulae Γ is satisfiable iff $\bigwedge_{\varphi \in \Gamma} \varphi$ is satisfiable. The induction hypothesis can be written as:

IH: for every node y in the subtree rooted at x_0 including x_0, if y is marked then Γ_y is not satisfiable.

Case 1. In the first case, it is not too hard to see that the satisfiability of Γ_x implies the satisfiability of Γ_{x_0} since the $\langle a \rangle$-rule preserves satisfiability from parent to child. By the induction hypothesis, we know that Γ_{x_0} is not satisfiable, hence Γ_x cannot be satisfiable.

Case 2. In the second case, we assume that Γ_{x_0} is satisfiable and derive a contradiction. We can then prove the claim as in the first case.

So, for a contradiction, let $M = (W, R, L)$ be a model and $w \in W$ a world such that (M, w) satisfies Γ_{x_0}: that is, $M, w \Vdash \chi$ for all $\chi \in \Gamma_{x_0}$. In particular, we have $M, w \Vdash \langle C \rangle \varphi$ by assumption since $\langle C \rangle \varphi \in x_0$. By definition of the semantics of $\langle C \rangle \varphi$, there must be a w-sequence σ of the form $w = \sigma_0, \sigma_1, \ldots$ and some $k \in \mathbb{N}_{>0}$ such that $M, \sigma_k \Vdash \varphi$. It is easy to see that we also have $M, \sigma_j \Vdash \langle C \rangle \varphi$ for all $j < k$.

Notation: For every $i \in \mathbb{N}$, let $b_i \in \mathrm{AG}$ be an agent such that $\sigma_i \, R_{b_i} \, \sigma_{i+1}$.

Our plan is to show $M, \sigma_j \not\Vdash \langle E \rangle \varphi$ for all $j < k$; in particular, we then have $M, \sigma_{k-1} \not\Vdash \langle E \rangle \varphi$ which contradicts our assumption that $M, \sigma_k \Vdash \varphi$ since it implies that $M, \sigma_{k-1} \Vdash [E] \neg \varphi$ and hence that $M, \sigma_k \Vdash \neg \varphi$. We do this by "walking down" the tableau T starting from x_0 and maintaining the following invariant for every node y which we meet:

Invariant: The node y is unmarked, the set of formulae Γ_y is satisfied by (M, w), and $\mathrm{uev}_y(\langle C \rangle \varphi) > h$.

If x_0 is an α-node, then its child y must be unmarked. Therefore, it must have $\mathrm{uev}_y(\langle C \rangle \varphi) > h$. Since all formulae of the form $\alpha \leftrightarrow \alpha_1 \wedge \alpha_2$ are valid and due to the definition of $[E]$, we also have that Γ_y is satisfied by (M, w) since Γ_{x_0} is satisfied by assumption.

If x_0 is a β-node, then it must have more than one child, and any marked child is unsatisfiable since it falls under the induction hypothesis. Moreover, due to the definition of $\langle E \rangle$, the fact that all formulae of the form $\beta \leftrightarrow \beta_1 \vee \beta_2$ are valid, and the fact that x_0 is satisfied by (M, w), there must exist at least one unmarked child y of x_0 such that Γ_y is satisfied by (M, w). Due to the construction of uev_{x_0}, this child has $\mathrm{uev}_y(\langle C \rangle \varphi) > h$.

We can repeat these steps until we arrive at a $\langle C \rangle$-node y which obeys the invariant and deals with $\langle C \rangle \varphi$. Note that this must happen before we encounter a state on our "walk". We also know that the first child y_1 of y must be marked, as otherwise, $\mathrm{uev}_y(\langle C \rangle \varphi)$ would be undefined by the $\langle C \rangle$-rule. Hence, we have already shown that the second child y_2 must obey the invariant. We also know by construction that y_2 contains $\langle E \rangle \langle C \rangle \varphi$. By the induction hypothesis, Γ_{y_1} is unsatisfiable. But as $\Gamma_{y_1} \setminus \{\langle E \rangle \varphi\}$ is a subset of Γ_y, which is satisfied by (M, w), this entails $M, w \not\Vdash \langle E \rangle \varphi$ and thus $M, \sigma_0 \not\Vdash \langle E \rangle \varphi$ since $\sigma_0 = w$.

If $0 = k-1$, we are done. Otherwise, we continue to "walk down" by repeating the steps as above from y_2 which contains $\langle E \rangle \langle C \rangle \varphi$.

On our way down, we must pass at least one $\langle E \rangle$-node z that obeys the invariant and deals with $\langle E \rangle \langle C \rangle \varphi$ before we reach a state. Of course, we could use the "normal" arguments for β-nodes to obtain *some* child of z that obeys the invariant, but for these nodes, we can even choose a specific child as shown next.

Let z_0 be the child of z that corresponds to the agent b_0. In the following, we show that z_0 obeys the invariant: as $0 < k-1$ (see above), we have $M, \sigma_1 \Vdash \langle C \rangle \varphi$ by our assumptions about σ. Together with $\sigma_0 \, R_{b_0} \, \sigma_1$ and $\sigma_0 = w$, this gives us $M, w \Vdash \langle b_0 \rangle \langle C \rangle \varphi$. Moreover, we have $\Gamma_{z_0} \setminus \{\langle b_0 \rangle \langle C \rangle \varphi\} \subseteq \Gamma_z$. Hence, Γ_{z_0} is satisfied by (M, w) as Γ_z is satisfied by (M, w). In particular, the node z_0 is unmarked because otherwise, it would be unsatisfiable due to the induction hypothesis. A glance at the definition of uev in the $\langle E \rangle$-rule reveals that this implies $\mathrm{uev}_{z_0}(\langle C \rangle \varphi) > h$, or else $\mathrm{uev}_z(\langle C \rangle \varphi)$ would be undefined. We conclude that z_0 indeed obeys the invariant and we can select it as successor of z on our "walk down".

Our choice of z_0 and the definitions of the rules imply that – additionally to the invariant – the following properties hold for every node y that appears after z on our "walk": the node y contains $\langle b_0\rangle\langle C\rangle\varphi$ and has $\mathrm{HAg}_y(\langle C\rangle\varphi) = b_0$.

Eventually, we arrive at an unmarked state s whose formulae Γ_s are satisfied by (M, w) and has $\mathrm{uev}_s(\langle C\rangle\varphi) > h$. As we must have passed at least one node that deals with $\langle E\rangle\langle C\rangle\varphi$, we also have $\langle b_0\rangle\langle C\rangle\varphi \in s$ and $\mathrm{HAg}_s(\langle C\rangle\varphi) = b_0$.

Next, we use the information about the state s to prove $M, \sigma_1 \nVdash \langle E\rangle\varphi$. Let x_1 be the (possibly virtual) successor of $\langle b_0\rangle\langle C\rangle\varphi \in s$. If x_1 is a virtual successor, a glance at the definition of uev_s in the $\langle a\rangle$-rule reveals that x_1 must lie on the path from x_0 to s (it could be x_0) as we have $\mathrm{uev}_s(\langle C\rangle\varphi) > h$ and $h := \mathrm{len}(\mathrm{HCr}_x)$ and $\mathrm{HAg}_s(\langle C\rangle\varphi) = b_0$. Thus, x_1 is unmarked and has $\mathrm{uev}_{x_1}(\langle C\rangle\varphi) > h$ as we have established these facts for all nodes on our way from x_0 down to s. If x_1 is a child of s (*i.e.* not a virtual successor), it follows directly from the facts about s and the definition of the $\langle a\rangle$-rule that x_1 is unmarked and has $\mathrm{uev}_{x_1}(\langle C\rangle\varphi) > h$. Another crucial point is that x_1 – whether a virtual successor or not – is still in the subtree rooted at x_0 in which the induction hypothesis holds.

What is more, we can deduce that (M, σ_1) satisfies Γ_{x_1} as follows. By definition of the $\langle a\rangle$-rule, Γ_{x_1} is of the form $\langle C\rangle\varphi \cup \Delta$ where $[b_0]\Delta \subseteq \Gamma_s$. We know $M, \sigma_1 \Vdash \langle C\rangle\varphi$ by assumption (remember $0 < k - 1$). We also know that (M, σ_0) in particular satisfies $[b_0]\Delta$ since we have established that $\Gamma_s \supseteq [b_0]\Delta$ is satisfied by (M, w) and $\sigma_0 = w$. As σ_1 is a b_0-successor world of σ_0 (*i.e.* $\sigma_0 \, R_{b_0} \, \sigma_1$), this implies that (M, σ_1) satisfies Δ, and hence Γ_{x_1}.

Putting together the properties of (M, σ_1) and x_1 reveals that we are in exactly the same position as for (M, σ_0) and x_0. Hence, we can "walk down" T from x_1, repeat all the arguments to maintain the invariant and derive $M, \sigma_1 \nVdash \langle E\rangle\varphi$.

Iterating the procedure (rather than setting up an inner induction) gives us that $M, \sigma_2 \nVdash \langle E\rangle\varphi, \ldots, M, \sigma_{k-1} \nVdash \langle E\rangle\varphi$, which concludes the proof of the second case. That is, by "walking down" the path from x_0, we have shown that no M can contain a world w and a w-sequence containing a world σ_k that satisfies φ. □

3.5 A Fully Worked Example

As an example, consider the statement "$(p \vee \neg p)$ is common knowledge" which can be written as the formula $[C](p \vee \neg p)$. This formula is obviously valid; hence its negation $\langle C\rangle(p \wedge \neg p)$ is not satisfiable and the root of an expanded tableau for $\langle C\rangle(p \wedge \neg p)$ should be marked.

Figure 1 and Figure 2 show such a tableau where the root node is node (1) in Figure 1 and where Figure 2 shows the sub-tableau rooted at node (4). Each node is classified as a ρ-node if rule ρ is applied to that node in the tableau. The edges labelled with $\langle 1\rangle$ and $\langle 2\rangle$ go from states to pre-states. Dotted frames indicate that the sub-tableaux at these nodes are not shown because they are very similar to sub-tableaux of other nodes: that is node $(5a)$ and $(4')$ are treated the same way as node $(1a)$ and (4), respectively. We have also omitted the history HAg as it does not play a role in this simple example. Dots "$\cdots$" indicate

that the corresponding values are not important for this example. The partial function uev maps $\langle C\rangle(p \wedge \neg p)$ to 1 and is undefined otherwise as explained below.

The marking of the nodes in Figure 1 with **true** is fairly straightforward, starting from the leaves (2b) and (2c) and flowing upwards, except for node (4). Our procedure constructs the tableau shown in Figure 2 for node (4). The leaf (6) is an $\langle a\rangle$-node, but it is "blocked" from creating the desired successor containing $\{\langle C\rangle(p \wedge \neg p)\}$ because there is a j such that $\mathrm{HCr}_6[j] = \{\langle C\rangle(p \wedge \neg p)\}$: namely $j = 1$. Thus the $\langle a\rangle$-rule computes $uev(\langle C\rangle(p \wedge \neg p)) = 1$ as stated above and also puts $mrk :=$ **false**. The leaf node (6') behaves in exactly the same way, so node (5b) does not change the value of uev in taking their minimum, and also computes $mrk =$ **false** $\vee$ **false** = **false**. Node (5a) has a dotted border since it behaves exactly as does node (1a), meaning that the β_1-child of node (5) is "closed". Since node (5b) is not "closed", node (5) inherits its uev and mrk values.

The crux of our procedure happens at node (4) which is an $\langle a\rangle$-node with $\mathrm{HCr}_4 = []$ and hence $len(\mathrm{HCr}_4) = 0$. The $\langle a\rangle$-rule therefore finds a child node (5) and a formula $\langle C\rangle(p \wedge \neg p)$ in it, which jointly satisfy $1 = uev_5(\langle C\rangle(p \wedge \neg p)) > len(\mathrm{HCr}_4) = 0$. That is, node (4) "sees" a child that "loops lower", meaning that node (5) is the root of an "isolated" subtree which does not fulfil its eventuality $\langle C\rangle(p \wedge \neg p)$. Thus the $\langle a\rangle$-rule sets $mrk_4 =$ **true**, marking (4) as "closed".

Returning to Figure 1, we find that the node (4') behaves in exactly the same way as does (4), which forces node (1b) to evaluate $mrk_{1b} =$ **true**. Since node (1) now has two "closed" β-children, it also gets marked as "closed".

3.6 The One-Pass Algorithm and its Complexity

Most tableau-based algorithms apply the rules in a particular order: namely, apply all the α- and β-rules until none are applicable, and then apply the $\langle a\rangle$-rule once. Of course, no rules are applied if the id-rule is applicable to close the branch. We have designed the rules so that they naturally capture this strategy, thereby giving a non-deterministic algorithm for constructing/traversing the tableau by just applying any one of the applicable rules. By fixing an arbitrary rule order and an arbitrary formula order, we can safely determinise this algorithm.

The use of histories and variables gives rise to an algorithm that constructs and traverses a tableau (deterministically) at the same time. On its way down the tableau, it constructs the set of formulae and the histories of a node by using information from the parent node; and on its way up, it synthesises the variables of a node according to the values of its children. Both steps are described by the rule that is applied to the node.

As soon as the algorithm has left a node on its way up, there is no need to keep the node in memory, it can safely be reclaimed as all important information has been passed up by the variables. Hence, the algorithm requires just one pass. Moreover, at any time, it only has to keep the current branch of the tableau in

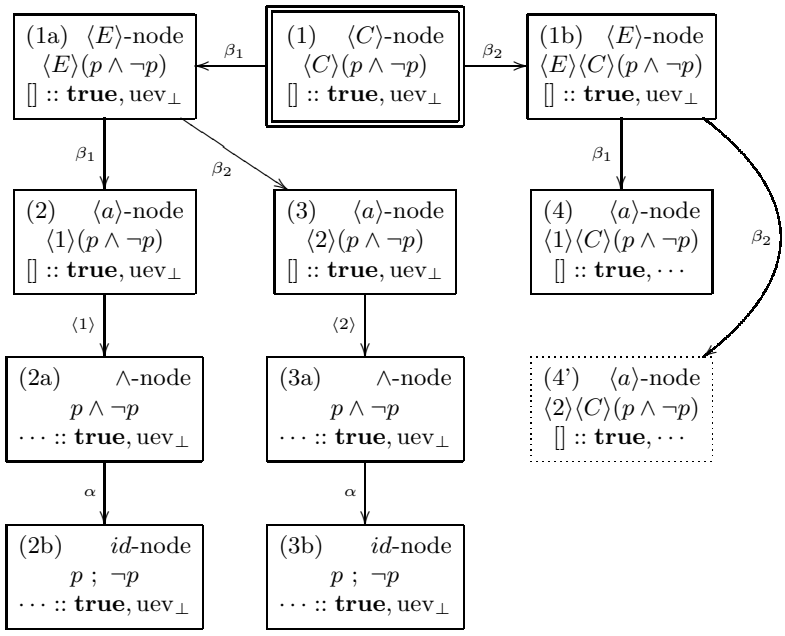

Fig. 1. An example: a tableau for $\langle C\rangle(p \wedge \neg p)$

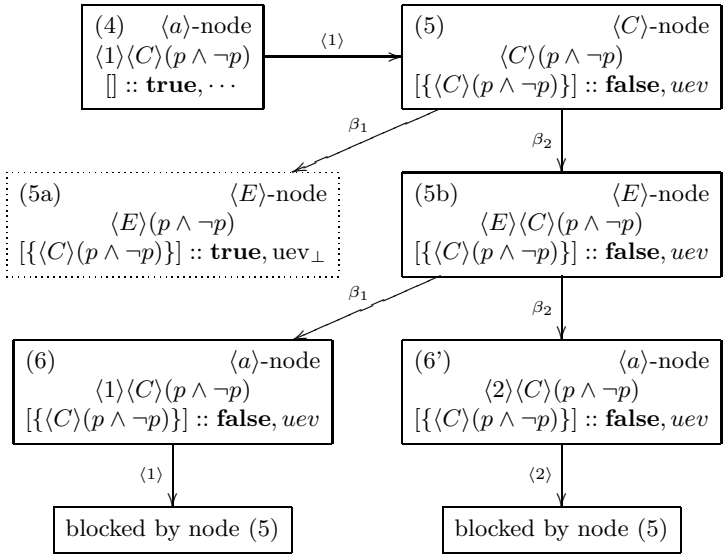

Fig. 2. An example: a tableau for $\langle C\rangle(p \wedge \neg p)$ (continued)

memory. The final result of the decision procedure can be obtained by looking at the variable mrk of the root which is the last node that has its variables set.

Of course, it is not always necessary to build the entire tableau. If, for example, the first child of an $\langle a \rangle$-rule is marked, the algorithm can mark the parent without having to look at the other children. Dually, if one child of a β-node is unmarked and has $\mathrm{uev}_\perp$ then there is no need to explore the other children since we can safely say that the parent is unmarked and has $\mathrm{uev}_\perp$. Other optimizations are possible and some of them are incorporated in our implementation in the Tableau Work Bench (`http://twb.rsise.anu.edu.au/twbdemo`), a generic tableau engine designed for rapid prototyping of (propositional) tableau calculi [1]. The high-level code of the prover for LCK – for two agents – is also visible there using the special input language designed for the TWB.

For analysing the complexity of our algorithm, we define the size $|\varphi|$ of a formula $\varphi \in \mathrm{Fml}$ as the number of symbols in φ where we regard $\langle a \rangle$, $[E]$, etc. as single symbols. To avoid indices, we set $k := N_{\mathrm{AG}}$.

Let $\phi \in \mathrm{Fml}$ be a formula in negation normal form of size $n := |\phi|$ and T a tableau for the node $(\{\phi\} :: \mathrm{HAg}_\perp, [] :: \cdots)$.

It is easy to see that $|\mathrm{cl}(\phi)| \in O(kn)$ and $|\varphi| \in O(n)$ for all $\varphi \in \mathrm{cl}(\phi)$. Hence, the number of different sets of formulae that can be assigned to the nodes of a tableau is in $2^{O(kn)}$. This means that the number of pre-states – and thus states – on a path in T is in $2^{O(kn)}$ too.

Proposition 24. *Let π be a path in T and let s_1 and s_2 be two (consecutive) states on π such that there is no state on π between s_1 and s_2. Then the number of nodes on π between s_1 and s_2 is in $O(n^2)$.*

Proof. We can ignore the *id*-rule because it can only be the last node of a path. Furthermore, we regard the special case first where the pre-state following s_1 in π contains only one formula φ.

The $\wedge$ and $\vee$-rules generate only proper subformulae of their principal formula; the $[E]$ and $\langle E \rangle$-rules generate only $\langle - \rangle$- and $[-]$-formulae which we can ignore because they are only considered by the $\langle a \rangle$-rule; the $[C]$ and $\langle C \rangle$-rules generate formulae that can only be processed by the $[E]$ or $\langle E \rangle$-rules. Therefore, there can only be $O(n)$ many such nodes in a row as the size of all formulae in $\mathrm{cl}(\phi)$ is in $O(n)$. Thus, we have – in the special case – that the number of nodes on π between s_1 and s_2 is in $O(n)$.

It is easy to see that the number of formulae in $\mathrm{cl}(\phi)$ which are not $\langle - \rangle$- or $[-]$-formulae is in $O(n)$. Hence, it is easy to see that the number of nodes on π between s_1 and s_2 is in $O(n) \cdot O(n) \subseteq O(n^2)$. □

As the number of states on a path is in $2^{O(kn)}$, and the number of nodes between two consecutive states is in $O(n^2)$, we have that the total number of nodes on a path is in $2^{O(kn)} \cdot O(n^2) \subseteq 2^{O(kn)}$.

The maximum degree in the tableau is clearly bounded by k and $\mathrm{cl}(\phi)$ and thus in $O(m)$ where $m := \max(k, n)$. Hence, the total number of nodes in T is in $O(m)^{2^{O(n)}} \subseteq 2^{2^{O(m)}}$. This follows from $n \leq m$ and the fact that, for $r, s \in \mathbb{N}_{>0}$:

$$(rm)^{2^{sm}} = 2^{\log_2((rm)^{2^{sn}})} = 2^{2^{sn} \cdot \log_2(rm)} \leq 2^{2^{(s+r)m}} \quad .$$

The time that the algorithm spends in a node is clearly dominated by the test whether the formula set of a pre-state is already in HCr. Due to the restriction on the length of a path, this can obviously be done in $2^{O(n)}$. Hence, our algorithm has a total running time in $2^{O(n)} \cdot 2^{2^{O(m)}} \subseteq 2^{2^{O(m)}}$.

Theorem 25. *For a fixed N_{AG}, the tableau algorithm for LCK outlined in this paper runs in double exponential deterministic time and needs exponential space.*

Proof. We have already shown that the algorithms runs in double exponential deterministic time. As our algorithm is a one-pass algorithm that does a traversal on the tree and stores information only in the nodes, the space that is needed depends on the length of a path; but we already know that every path in the tableau has at most exponential length, which concludes the proof. □

References

1. P. Abate. The Tableau Workbench: a framework for building automated tableau-based theorem provers. PhD thesis, The Australian National University, 2006.
2. L. Alberucci. The Modal μ-Calculus and the Logic of Common Knowledge. PhD thesis, Universität Bern, 2002.
3. A. Basukoski and A. Bolotov. A clausal resolution method for branching time logic ECTL+. *Annals of Mathematics and Artificial Intelligence*, 46(3):235–263, 2006.
4. M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. In *Proceedings of Principles of Programming Languages*, 1981.
5. J. Bradfield and C. Stirling. Modal logics and mu-calculi. Handbook of Process Algebra, pages 293–332. Elsevier Science Publishers, Amsterdam, 2001.
6. C. Dixon and M. Fisher. Resolution-based proof for multi-modal temporal logics of knowledge. In *Proc. TIME 2000*, pages 69-78.
7. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. Reasoning about Knowledge. The MIT Press, Cambridge, Massachusetts, 1995.
8. G. Gough. Decision procedures for temporal logics. Master's thesis, Dept. of Computer Science, University of Manchester, England, 1984.
9. J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. In *Artificial Intelligence 54*, pages 319–379, 1992.
10. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. LPAR'99*, LNCS 1705:161–180. Springer, 1999.
11. I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
12. U. Hustadt and B. Konev. TRP++: A temporal resolution prover. In *Collegium Logicum*, pages 65–79. Kurt Gödel Society, 2004.
13. G. Jäger, M. Kretz, and T. Studer. Cut-free common knowledge. To appear `http://www.iam.unibe.ch/til/publications/to_appear/jaeger_kretz_studer_06`
14. M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer Systems and Science*, 1986.