ELSEVIER

Available online at www.sciencedirect.com **Electronic Notes in
Theoretical Computer
Science**

Electronic Notes in Theoretical Computer Science 231 (2009) 191–209

www.elsevier.com/locate/entcs

An On-the-fly Tableau-based Decision Procedure for *PDL*-satisfiability

Pietro Abate^{a,1}, Rajeev Goré^{a,2} and Florian Widmann^{b,3,4}^a Computer Sciences Laboratory, The Australian National University, Canberra, Australia^b Computer Sciences Laboratory and Logic & Computation Programme
The Australian National University and NICTA, Canberra, Australia

Abstract

We present a tableau-based algorithm for deciding satisfiability for propositional dynamic logic (*PDL*) which builds a finite rooted tree with ancestor loops and passes extra information from children to parents to separate good loops from bad loops during backtracking. It is easy to implement, with potential for parallelisation, because it constructs a pseudo-model “on the fly” by exploring each tableau branch independently. But its worst-case behaviour is 2EXPTIME rather than EXPTIME. A prototype implementation in the TWB (<http://twb.rsise.anu.edu.au>) is available.

Keywords: propositional dynamic logic, automated reasoning, tableau calculus, decision procedure

1 Introduction

Propositional dynamic logic (*PDL*) is a logic for reasoning about programs [14,9]. Its formulae consist of traditional Boolean formulae plus “action modalities” built from a finite set of atomic programs using sequential composition (;), non-deterministic choice ($\cup$), repetition (*), and test (?). The satisfiability problem for *PDL* is EXPTIME-complete [15]. Unlike EXPTIME-complete description logics with algorithms exhibiting good average-case behaviour, no decision procedures for *PDL*-satisfiability are satisfactory from both a theoretical (soundness and completeness) and practical (average case behaviour) viewpoint as we explain below.

The earliest decision procedures for *PDL* are due to Fischer and Ladner [9] and Pratt [15]. Fischer and Ladner’s method is impractical because it first constructs

¹ Email: Pietro.Abate@pps.jussieu.fr

² Email: Rajeev.Gore@rsise.anu.edu.au

³ Email: Florian.Widmann@rsise.anu.edu.au

⁴ National ICT Australia is funded by the Australian Government’s Dept 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 set of all consistent subsets of the set of all subformulae of the given formula, which always requires exponential time in all cases. On the other hand, Pratt [15] essentially builds a multi-pass (explained shortly) tableau method. Most subsequent decision procedures for other fix-point logics like propositional linear temporal logic (PLTL) [18] and computation tree logic (CTL) [5,8] trace back to Pratt’s procedure for PDL [15], and they all share one main disadvantage as explained next.

In these multi-pass procedures, the first pass constructs a rooted tableau of nodes containing formula-sets, but allows cross-branch arcs from a node n on one branch to a (previously constructed) node m on a different branch if applying the tableau construction to n would duplicate m . The result is a potentially exponential-sized cyclic graph (rather than a cyclic tree where m would have to be an ancestor of n). The cyclic graph is a “pseudo-model” because it may not fulfil the requirement of diamond-like formulae (“eventualities”) that “a certain formula is eventually true”. The subsequent passes check that the “pseudo-model” is a real model by pruning inconsistent nodes and pruning nodes containing “unfulfilled eventualities”.

Although efficient model-checking techniques can check the “pseudo-model” in time which is linear in its size, these multi-pass methods can construct an exponential-sized cyclic graph needlessly. One solution is to check for fulfilled eventualities “on the fly”, as the graph is built, and although such methods exist for model-checking [7,6], we know of no such decision procedures for *PDL*. The only implementation of a multiple-pass method for *PDL* that we know of is in LoTRec (www.irit.fr/Lotrec) but it is not optimal as it treats disjunctions naively.

Baader [4] gave a single-pass tableau-based decision procedure for a description logic with role definitions involving union, composition and transitive closure of roles: essentially *PDL* without test. His method constructs a (cyclic tree) tableau using the semantics of the *PDL* operators. To separate “good loops” from “bad loops”, Baader must decide equality of regular languages, a PSPACE-complete problem which in practice may require exponential time. Instead of solving these problems “on the fly”, they can be reduced to a simple check on the identity of states in a deterministic minimal automaton created from the positive regular expressions appearing in the initial formula during a pre-processing stage [4, page 27]. But since the pre-computed automaton can be of exponential size, this alternative may require exponential time needlessly. Baader’s method is double-exponential in the worst-case. The “test” construct is essential to express “while” loops but creates a mutual recursion between the Boolean language and the regular language. It is not obvious to us how to extend Baader’s method to “test”. DLP (<http://www.cs.bell-labs.com/cm/cs/who/pfps/dlp>) implements this method restricted to test-free formulae where $*$ applies only to atomic programs.

De Giacomo and Massacci [10] gave an optimal *PDL*-satisfiability test using labelled formulae like $\sigma : \varphi$ to capture that “possible world σ makes formula φ true”. They first give an NEXPTIME algorithm for deciding *PDL*-satisfiability and then discuss ways to obtain an EXPTIME version using various known results. But an actual EXPTIME algorithm, and its soundness and completeness proofs, are not given. A deterministic implementation of their NEXPTIME algorithm by Schmidt

and Tishkovsky struck problems with nested stars, but a solution is forthcoming [16].

Other decision procedures for fix-point logics use resolution calculi, translation methods, automata-theoretic methods, and game theoretic methods: see [1] for references. We know of no implementations for *PDL* based on these methods.

Here, we give a sound, complete and terminating decision procedure for *PDL* with the following advantages and disadvantages:

One-pass nature: our method constructs a single-rooted finite tree (with loops from leaves to ancestors). As there are no cross-branch edges, we can use depth-first, left-to-right search, reclaiming the space used for each branch when backtracking.

Proofs: Full elementary proofs of soundness and completeness are available [2].

Ease of implementation: our rules are easy to implement since our tableau nodes contain sets of formulae and some easily defined extra information whose manipulation requires only basic operations on sets and integers. However, these low-level details make the rules cumbersome to describe.

Potential for optimisation: there is potential to optimise our (tree) tableaux using successful techniques from (one-pass) tableaux for description logics [12].

Ease of generating counter-models: the soundness proof immediately gives an effective procedure for turning an “open” tableau into a *PDL*-model.

Ease of generating proofs: unlike existing Gentzen calculi for fix-point logics [3,13], our tableau calculus gives a cut-free Gentzen-style calculus with “cyclic proofs” with an optimal rather than worst-case bound for the finitised omega rule.

Potential for parallelisation: our rules build the branches independently but combine their results during backtracking, enabling a parallel implementation.

Prototype: a (sequential) prototype implementation in the Tableau Work Bench (twb.rsise.anu.edu.au) allows to test arbitrary *PDL* formulae over the web.

Complexity: our method has worst-case double-exponential time complexity.

Generality: Our method for *PDL* fits into a class of similar “one pass” methods for other fix-point logics like PLTL [17] and CTL [1]. Further experimental work is required to determine if our methods can be optimised to exhibit good average-case behaviour using techniques like sound global caching [11].

2 Syntax, Semantics and Hintikka Structures

Definition 2.1 Let AFml and APrg be two disjoint and countably infinite sets of propositional atoms and *atomic programs*, respectively. The set Fml of all formulae and the set Prg of all *programs* are defined inductively as follows:

- (i) $\text{AFml} \subseteq \text{Fml}$ and $\text{APrg} \subseteq \text{Prg}$
- (ii) if $\varphi, \psi \in \text{Fml}$ then $\neg\varphi \in \text{Fml}$ and $\varphi \wedge \psi \in \text{Fml}$ and $\varphi \vee \psi \in \text{Fml}$ and $\varphi? \in \text{Prg}$
- (iii) if $\varphi \in \text{Fml}$ and $\alpha \in \text{Prg}$ then $\langle\alpha\rangle\varphi \in \text{Fml}$ and $[\alpha]\varphi \in \text{Fml}$
- (iv) if $\alpha \in \text{Prg}$ and $\beta \in \text{Prg}$ then $(\alpha; \beta) \in \text{Prg}$ and $\alpha \cup \beta \in \text{Prg}$ and $\alpha^* \in \text{Prg}$.

A $\langle \rangle$ -formula is any formula $\langle \alpha \rangle \varphi$, a $\langle q \rangle$ -formula is a $\langle \rangle$ -formula $\langle \alpha \rangle \varphi$ with $\alpha \notin \text{APrg}$, and a $\langle * \rangle$ -formula is any formula $\langle \alpha * \rangle \varphi$. $\text{Fml}\langle \rangle$ is the set of all $\langle \rangle$ -formulae, $\text{Fml}\langle q \rangle$ is the set of all $\langle q \rangle$ -formulae, and $\text{Fml}\langle * \rangle$ is the set of all $\langle * \rangle$ -formulae. Implication ($\rightarrow$), equivalence ($\leftrightarrow$) and the constants Falsum and Verum are not part of the core language but can be defined as usual.

In the rest of the article, let p, q range over members of AFml and a, b range over members of APrg .

Definition 2.2 A *transition frame* is a pair (W, R) where W is a non-empty set of worlds and R a function that maps each atomic program a to a binary relation R_a over W . A *model* (W, R, V) is a transition frame (W, R) and a valuation function $V : \text{AFml} \rightarrow 2^W$ mapping each atomic proposition p to a set $V(p)$ of worlds.

Definition 2.3 Let $M = (W, R, V)$ be a model. The functions $\tau_M : \text{Fml} \rightarrow 2^W$ and $\rho_M : \text{Prg} \rightarrow 2^{W \times W}$ are defined inductively as follows:

$$\begin{aligned}
 \tau_M(p) &:= V(p) & \rho_M(a) &:= R_a & \tau_M(\neg \varphi) &:= W \setminus \tau_M(\varphi) \\
 \tau_M(\varphi \wedge \psi) &:= \tau_M(\varphi) \cap \tau_M(\psi) & \tau_M(\varphi \vee \psi) &:= \tau_M(\varphi) \cup \tau_M(\psi) \\
 \tau_M([\alpha] \varphi) &:= \{w \mid \forall v \in W. (w, v) \in \rho_M(\alpha) \Rightarrow v \in \tau_M(\varphi)\} \\
 \tau_M(\langle \alpha \rangle \varphi) &:= \{w \mid \exists v \in W. (w, v) \in \rho_M(\alpha) \ \& \ v \in \tau_M(\varphi)\} \\
 \rho_M(\alpha \cup \beta) &:= \rho_M(\alpha) \cup \rho_M(\beta) & \rho_M(\varphi?) &:= \{(w, w) \mid w \in \tau_M(\varphi)\} \\
 \rho_M(\alpha; \beta) &:= \{(w, v) \mid \exists u \in W. (w, u) \in \rho_M(\alpha) \ \& \ (u, v) \in \rho_M(\beta)\} \\
 \rho_M(\alpha*) &:= \{(w, v) \mid \exists k \in \mathbb{N}. \exists w_0, \dots, w_k \in W. (w_0 = w \ \& \ w_k = v \ \& \\
 &\quad \forall i \in \{0, \dots, k-1\}. (w_i, w_{i+1}) \in \rho_M(\alpha))\}
 \end{aligned}$$

For $w \in W$ and $\varphi \in \text{Fml}$, we write $M, w \Vdash \varphi$ iff $w \in \tau_M(\varphi)$.

Definition 2.4 Formula $\varphi \in \text{Fml}$ is *satisfiable* iff there is a model $M = (W, R, V)$ and a $w \in W$ such that $M, w \Vdash \varphi$. Formula $\varphi \in \text{Fml}$ is *valid* iff $\neg \varphi$ is not satisfiable.

Definition 2.5 Formula $\varphi \in \text{Fml}$ is in *negation normal form* if $\neg$ appears only immediately before propositional atoms. For every $\varphi \in \text{Fml}$, we obtain a formula $\text{nnf}(\varphi)$ in negation normal form by pushing negations inward repeatedly (e.g. using de Morgan's laws) so $\varphi \leftrightarrow \text{nnf}(\varphi)$ is valid. We define $\sim \varphi := \text{nnf}(\neg \varphi)$.

We use Smullyan's α/β -notation to categorise formulae via Table 1 and use bolding to differentiate it from the use of α and β as members of Prg . So if α (respectively β) is any formula pattern in the first row then α_1 and α_2 (respectively β_1 and β_2) are its corresponding patterns in the second and third row.

Proposition 2.6 All formulae $\alpha \leftrightarrow \alpha_1 \wedge \alpha_2$ and $\beta \leftrightarrow \beta_1 \vee \beta_2$ in Table 1 are valid.

Definition 2.7 A *structure* (W, R, L) [for $\varphi \in \text{Fml}$] is a transition frame (W, R) and a labelling function $L : W \rightarrow 2^{\text{Fml}}$ which associates with each world $w \in W$ a set $L(w)$ of formulae [and has $\varphi \in L(v)$ for some world $v \in W$].

Table 1
Smullyan's α - and β -notation to classify formulae

α	$\varphi \wedge \psi$	$[\alpha \cup \beta]\varphi$	$[\alpha*]\varphi$	$\langle\psi?\rangle\varphi$	$\langle\alpha; \beta\rangle\varphi$	$[\alpha; \beta]\varphi$
α_1	φ	$[\alpha]\varphi$	φ	φ	$\langle\alpha\rangle\langle\beta\rangle\varphi$	$[\alpha][\beta]\varphi$
α_2	ψ	$[\beta]\varphi$	$[\alpha][\alpha*]\varphi$	ψ		
	β	$\varphi \vee \psi$	$\langle\alpha \cup \beta\rangle\varphi$	$\langle\alpha*\rangle\varphi$	$[\psi?]\varphi$	
	β_1	φ	$\langle\alpha\rangle\varphi$	φ	φ	
	β_2	ψ	$\langle\beta\rangle\varphi$	$\langle\alpha\rangle\langle\alpha*\rangle\varphi$	$\sim\psi$	

Definition 2.8 For a given $\varphi \in \text{Fml}$ the (infinite) set $\text{pre}(\varphi)$ is defined as:

$$\text{pre}(\varphi) := \{\psi \in \text{Fml} \mid \exists k \in \mathbb{N}. \exists \alpha_1, \dots, \alpha_k \in \text{Prg}. \psi = \langle\alpha_1\rangle \dots \langle\alpha_k\rangle\varphi\}.$$

For all formulae φ and ψ , the binary relation $\rightsquigarrow$ on formulae is defined as: $\varphi \rightsquigarrow \psi$ iff (exactly) one of the following conditions is true:

- $\exists \chi \in \text{Fml}. \exists \alpha, \beta \in \text{Prg}. \varphi = \langle\alpha; \beta\rangle\chi$ & $\psi = \langle\alpha\rangle\langle\beta\rangle\chi$
- $\exists \chi \in \text{Fml}. \exists \alpha, \beta \in \text{Prg}. \varphi = \langle\alpha \cup \beta\rangle\chi$ & ($\psi = \langle\alpha\rangle\chi$ or $\psi = \langle\beta\rangle\chi$)
- $\exists \chi \in \text{Fml}. \exists \alpha \in \text{Prg}. \varphi = \langle\alpha*\rangle\chi$ & ($\psi = \chi$ or $\psi = \langle\alpha\rangle\langle\alpha*\rangle\chi$)
- $\exists \chi, \phi \in \text{Fml}. \varphi = \langle\phi?\rangle\chi$ & $\psi = \chi$.

Intuitively, using Table 1, the “ $\rightsquigarrow$ ” relates a $\langle\mathcal{A}\rangle$ -formulae α (respectively β), to α_1 (respectively β_1 and β_2) while $\text{pre}(\varphi)$ captures that $\langle\alpha*\rangle\varphi$ can be “reduced” to $\langle\alpha\rangle\langle\alpha*\rangle\varphi$, which can be reduced to $\langle\alpha_1\rangle \dots \langle\alpha_k\rangle\langle\alpha*\rangle\varphi$. Note that $\varphi \in \text{pre}(\varphi)$.

Definition 2.9 Let $H = (W, R, L)$ be a structure, $\varphi \in \text{Fml}$ a formula, $\beta \in \text{Prg}$ a program, and $w \in W$ a state. A *fulfilling chain* for (φ, β, w) in H is a finite sequence $(w_0, \psi_0), \dots, (w_n, \psi_n)$ of world-formula pairs with $n \geq 0$ such that:

- $w_i \in W$, $\psi_i \in \text{pre}(\varphi)$, and $\psi_i \in L(w_i)$ for all $0 \leq i \leq n$
- $w_0 = w$, $\psi_0 = \langle\beta\rangle\varphi$, $\psi_n = \varphi$, and $\psi_i \neq \varphi$ for all $0 \leq i \leq n-1$
- for all $0 \leq i \leq n-1$, if $\psi_i = \langle a \rangle\chi$ for some $a \in \text{APrg}$ and $\chi \in \text{Fml}$ then $\psi_{i+1} = \chi$ and $w_i R_a w_{i+1}$; otherwise $\psi_i \rightsquigarrow \psi_{i+1}$ and $w_i = w_{i+1}$.

Each ψ_i is in $L(w_i)$, the chain starts at $(w_0, \langle\beta\rangle\varphi)$, ends at (w_n, φ) , and no other w_i is paired with φ . Formulae ψ_i, ψ_{i+1} are $\rightsquigarrow$ -related and corresponding worlds w_i, w_{i+1} are equal unless $\psi_i = \langle a \rangle\chi$, in which case $\psi_{i+1} = \chi$ and $w_i R_a w_{i+1}$. Thus eventuality $\langle\beta\rangle\varphi \in w_0$ is fulfilled by $\varphi \in w_n$ and w_n is β -reachable from w_0 .

Definition 2.10 A *pre-Hintikka structure* $H = (W, R, L)$ [for $\varphi \in \text{Fml}$] is a structure [for φ] that satisfies H1-H5 (below) for every $w \in W$ where α and β are formulae as defined in Table 1. A *Hintikka structure* $H = (W, R, L)$ [for $\varphi \in \text{Fml}$]

is a pre-Hintikka structure [for φ] that additionally satisfies H6 below:

$$\text{H1} : \neg p \in L(w) \Rightarrow p \notin L(w)$$

$$\text{H2} : \alpha \in L(w) \Rightarrow \alpha_1 \in L(w) \ \& \ \alpha_2 \in L(w)$$

$$\text{H3} : \beta \in L(w) \Rightarrow \beta_1 \in L(w) \text{ or } \beta_2 \in L(w)$$

$$\text{H4} : \langle a \rangle \varphi \in L(w) \Rightarrow \exists v \in W. w R_a v \ \& \ \varphi \in L(v)$$

$$\text{H5} : [a] \varphi \in L(w) \Rightarrow \forall v \in W. w R_a v \Rightarrow \varphi \in L(v)$$

$$\text{H6} : \langle \alpha * \rangle \varphi \in L(w) \Rightarrow \text{there exists a fulfilling chain for } (\varphi, \alpha *, w) \text{ in } H.$$

H3 “locally unwinds” the fix-point semantics of $\langle \alpha * \rangle \varphi$, but does not guarantee a *least* fix-point which requires φ be true eventually. H6 “globally” ensures all $\langle * \rangle$ -formulae are fulfilled. H2 captures the *greatest* fix-point semantics of $[\alpha *] \varphi$.

Theorem 2.11 *A formula $\varphi \in \text{Fml}$ in negation normal form is satisfiable iff there exists a Hintikka structure for φ . (See [2] for a proof).*

3 An Overview of the Algorithm

To track unfulfilled eventualities and to avoid “at a world” cycles, our algorithm stores additional information in each tableau node using *histories* and *variables* [17]. Histories are passed from parents to children and variables from children to parents.

Our algorithm starts at a root containing a given formula ϕ and some default history values. It builds a tree by repeatedly applying α -/ β -rules to decompose formulae via the semantics of *PDL*. The β -rule for $\langle \alpha * \rangle \varphi$ has a left child that fulfils this eventuality by reducing it to φ , and a right child that procrastinates fulfilment by “reducing” it to $\langle \alpha \rangle \langle \alpha * \rangle \varphi$. The rules modify the histories and variables as appropriate for their intended purpose.

But naive application of the α -/ β -rules to formulae like $\langle a * * \rangle \varphi$ with nested stars can lead to “at a world” cycles: e.g. $\langle a * * \rangle \varphi \rightsquigarrow \langle a * \rangle \langle a * * \rangle \varphi \rightsquigarrow \langle a * * \rangle \varphi$. A solution is to use the histories to reduce one particular $\langle \alpha \rangle$ -formula until α becomes atomic by forcing the rules to concentrate on this task, and to block previously reduced diamonds and boxes if they lead to “at a world” cycles. The application of α -/ β -rules stops when all non-blocked leaves contain only atoms, negated atoms, and all $\langle \rangle$ -formulae and all $[]$ -formulae begin with outermost atomic programs only.

For each such leaf node l , and for each such formula $\langle b \rangle \xi$ in l , the $\langle \rangle$ -rule creates a successor node containing $\{\xi\} \cup \Delta$, where $\Delta = \{\psi \mid [b]\psi \in l\}$. These successors are then saturated to produce new leaves using the α - and β -rules, and the $\langle \rangle$ -rule creates the successors of these new leaves, and so on.

If left unchecked, this procedure can produce infinite branches since the same successors can be created again and again on the same branch. To obtain termination, the $\langle \rangle$ -rule creates a successor containing $\{\xi\} \cup \Delta$ for l only if this successor has not already been created previously higher up on the current branch.

So if the successor $\{\xi\} \cup \Delta$ exists already, the current branch is “blocked” from re-creating it. The resulting loop may be “bad” since every β -node on this branch for an eventuality $\langle \alpha^* \rangle \varphi$ may procrastinate, so $\langle \alpha^* \rangle \varphi$ is never fulfilled. To track this potentially unfulfilled eventuality, we assign the height of the blocking node to the pair $(\xi, \langle \alpha^* \rangle \varphi)$ via a variable uev as long as ξ is a decomposition of $\langle \alpha^* \rangle \varphi$.

During backtracking, our rules “merge” the uev entries of the children and also modify the resulting uev to reverse-track the decomposition of $\langle \alpha^* \rangle \varphi$. In particular, a uev entry becomes undefined at a node if the eventuality it tracks can be fulfilled in the sub-tableau rooted at this node. Conversely, if a node at height h receives a uev entry with value at least h then the eventuality tracked by this uev entry definitely cannot be fulfilled, so the parent of this (blocking) node is then unsatisfiable. The initial formula ϕ is satisfiable iff the root has status “open”. Due to “at a world” cycles, the status can be “unsatisfiable”, “open” or “barred” (explained later).

4 A One-pass Tableau Algorithm for PDL

Definition 4.1 A *tableau node* x is of the form $(\Gamma :: \text{HCr}, \text{Nx}, \text{BD}, \text{BB} :: \text{stat}, \text{uev})$ where: Γ is a set of formulae; HCr is a list of pairs (φ, Δ) where Δ is a set of formulae and $\varphi \in \Delta$; Nx is either $\perp$ or a formula designated to be the principal formula of the rule applied to x ; BD is the set of “Blocked Diamonds”; BB is the set of “Blocked Boxes”; stat has one of the values **unsat**, **open**, or **barred**; and uev is a partial function from $\text{Fml}\langle \rangle \times \text{Fml}\langle * \rangle$ to $\mathbb{N}_{>0}$ (the positive natural numbers).

Definition 4.2 A *tableau* for a formula set $\Gamma \subseteq \text{Fml}$ and histories HCr, Nx, BD, and BB is a tree of tableau nodes with root $(\Gamma :: \text{HCr}, \text{Nx}, \text{BD}, \text{BB} :: \text{stat}, \text{uev})$ 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) but where the parent can inherit some information from the children. A tableau is *expanded* if every node has (exactly) one rule applied to it. On any branch of a 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 .

The list HCr is a history for detecting ancestor-loops and guarantees termination. The choice of principal formula is free if $\text{Nx} = \perp$, but is pre-determined as the formula in Nx otherwise. When a diamond formula in the parent is decomposed to give a formula $\varphi \in \text{Fml}\langle \varphi \rangle$ in the child node, we set the Nx-value of the child to φ to ensure that φ is decomposed next. Together with the histories BD and BB, this allows us to block $\langle \alpha^* \rangle$ -formulae and $[\alpha^*]$ -formulae from creating “at a world” cycles. The variables stat and uev have their values determined by the children of a node. Formally, $\text{stat} = \text{unsat}$ at node x if x is definitely unsatisfiable. Informally, $\text{stat} = \text{barred}$ if all descendants of node x are unsatisfiable or lead to an “at a world” cycle. Finally, $\text{stat} = \text{open}$ indicates that the node is potentially satisfiable, but as it may be on a loop, this is something which we can determine only later as we backtrack towards the root.

Definition 4.3 The partial function $\text{uev}_\perp : \text{Fml}\langle \rangle \times \text{Fml}\langle * \rangle \rightarrow \mathbb{N}_{>0}$ is the constant function that is undefined for all pairs of formulae: i.e. $\forall \psi_1, \psi_2. \text{uev}_\perp(\psi_1, \psi_2) = \perp$.

The partial functions $\text{Nxtst} : \text{Fml} \rightarrow \text{Fml}$ and $\text{BDtst} : \text{Fml} \times 2^{\text{Fml}} \rightarrow 2^{\text{Fml}}$ are:

$$\text{Nxtst}(\chi) := \begin{cases} \chi & \text{if } \chi \in \text{Fml}\langle\varphi\rangle \\ \perp & \text{otherwise} \end{cases} \quad \text{BDtst}(\chi, \Gamma) := \begin{cases} \Gamma & \text{if } \chi \in \text{Fml}\langle\varphi\rangle \\ \emptyset & \text{otherwise.} \end{cases}$$

The function Nxtst returns $\perp$ when the formula being tested is not a $\langle\varphi\rangle$ -formula, or is a $\langle\varphi\rangle$ -formula but its program is atomic. The function uev tracks unfulfilled eventualities: if $\text{uev}(\chi_1, \chi_2) = h \neq \perp$, the potentially unfulfilled eventuality related to χ_1 and χ_2 will be resolved when we backtrack to “height” h . If a node has $\text{stat} = \mathbf{unsat}$ or $\text{stat} = \mathbf{barred}$ then its uev is irrelevant so it is arbitrarily set to $\text{uev}_\perp$.

4.1 The Rules

We use Γ and Δ for sets of formulae and write $\varphi_1, \dots, \varphi_n, \Delta_1, \dots, \Delta_m$ for the partition $\{\varphi_1\} \uplus \dots \uplus \{\varphi_n\} \uplus \Delta_1 \uplus \dots \uplus \Delta_m$ of formulae in a node. To save space, we often omit histories/variables which are passed unchanged from parents/children to children/parents. Most rules are applicable only if some side-conditions hold, and most involve actions that change histories downwards or variables upwards. See Section 4.4 for two examples.

Terminal Rules.

$$(id) \quad \frac{(\Gamma :: \dots :: \text{stat}, \text{uev})}{\{p, \neg p\} \subseteq \Gamma \text{ for some } p \in \text{AFml}}$$

Actions for (id) : $\text{stat} := \mathbf{unsat}$ and $\text{uev} := \text{uev}_\perp$.

$$(\langle*\rangle_2) \quad \frac{(\langle\alpha*\rangle\varphi, \Gamma :: \text{Nx}, \text{BD} :: \text{stat}, \text{uev})}{\text{Nx} \in \{\perp, \langle\alpha*\rangle\varphi\} \ \& \ \langle\alpha*\rangle\varphi \in \text{BD}}$$

Actions for $(\langle*\rangle_2)$: $\text{stat} := \mathbf{barred}$ and $\text{uev} := \text{uev}_\perp$.

An id -node is clearly unsatisfiable. Since the principal formula of the $\langle*\rangle_2$ -rule is in BD, it causes an “at a world” cycle, so this rule terminates the current branch. Note that both rules may be applicable to a node.

Linear (α) Rules.

$$\begin{array}{ll} (\wedge) \quad \frac{(\varphi \wedge \psi, \Gamma :: \text{Nx} :: \text{uev})}{(\varphi, \psi, \Gamma :: \text{Nx} :: \text{uev}_1)} & ([\cup]) \quad \frac{([\alpha \cup \beta]\varphi, \Gamma :: \text{Nx} :: \text{uev})}{([\alpha]\varphi, [\beta]\varphi, \Gamma :: \text{Nx} :: \text{uev}_1)} \\ ([;]) \quad \frac{([\alpha; \beta]\varphi, \Gamma :: \text{Nx} :: \text{uev})}{([\alpha][\beta]\varphi, \Gamma :: \text{Nx} :: \text{uev}_1)} & ([*]) \quad \frac{([\alpha*]\varphi, \Gamma :: \text{Nx}, \text{BB} :: \text{uev})}{(\Gamma_1 :: \text{Nx}, \text{BB}_1 :: \text{uev}_1)} \end{array}$$

Common Side Condition: $\text{Nx} = \perp$.

Common Action: $\text{uev}(\chi_1, \chi_2) := \text{if } \chi_1 \in \Gamma \text{ then } \text{uev}_1(\chi_1, \chi_2) \text{ else } \perp$

Extra Actions for $([*])$:

$$\begin{aligned}\Gamma_1 &:= \text{if } [\alpha*]\varphi \in \text{BB} \text{ then } \Gamma \text{ else } \{\varphi\} \cup \{[\alpha][\alpha*]\varphi\} \cup \Gamma \\ \text{BB}_1 &:= \{[\alpha*]\varphi\} \cup \text{BB}\end{aligned}$$

Most rules are standard but for the histories since they just capture the transformations in Table 1. The $[*]$ -rule just deletes $[\alpha*]\varphi$ if $[\alpha*]\varphi \in \text{BB}$ since this indicates that it has already been expanded once “at this world”. Otherwise it captures the fix-point nature of $[\alpha*]\varphi$ via Prop. 2.6 and then puts $[\alpha*]\varphi$ into BB_1 .

The next two rules have individual side-conditions and actions as shown.

$$(\langle; \rangle) \quad \frac{(\langle\alpha; \beta\rangle\varphi, \Gamma :: \text{Nx}, \text{BD} :: \text{uev})}{(\langle\alpha\rangle\langle\beta\rangle\varphi, \Gamma :: \text{Nx}_1, \text{BD}_1 :: \text{uev}_1)} \quad \text{Nx} \in \{\perp, \langle\alpha; \beta\rangle\varphi\}$$

Actions for $(\langle; \rangle)$:

$$\begin{aligned}\text{Nx}_1 &:= \text{Nxtst}(\langle\alpha\rangle\langle\beta\rangle\varphi) \\ \text{BD}_1 &:= \text{BDtst}(\langle\alpha\rangle\langle\beta\rangle\varphi, \text{BD}) \\ \text{uev}(\chi_1, \chi_2) &:= \begin{cases} \text{uev}_1(\langle\alpha\rangle\langle\beta\rangle\varphi, \chi_2) & \text{if } \chi_1 = \langle\alpha; \beta\rangle\varphi \\ \text{uev}_1(\chi_1, \chi_2) & \text{if } \chi_1 \in \Gamma \\ \perp & \text{otherwise} \end{cases}\end{aligned}$$

$$(\langle?\rangle) \quad \frac{(\langle\psi?\rangle\varphi, \Gamma :: \text{Nx}, \text{BD} :: \text{uev})}{(\langle\psi, \varphi, \Gamma :: \text{Nx}_1, \text{BD}_1 :: \text{uev}_1)} \quad \text{Nx} \in \{\perp, \langle\psi?\rangle\varphi\}$$

Actions for $(\langle?\rangle)$:

$$\begin{aligned}\text{Nx}_1 &:= \text{Nxtst}(\varphi) \\ \text{BD}_1 &:= \text{BDtst}(\varphi, \text{BD}) \\ \text{uev}(\chi_1, \chi_2) &:= \begin{cases} \text{uev}_1(\varphi, \chi_2) & \text{if } \chi_1 = \langle\psi?\rangle\varphi \\ \text{uev}_1(\chi_1, \chi_2) & \text{if } \chi_1 \in \Gamma \\ \perp & \text{otherwise} \end{cases}\end{aligned}$$

These rules just capture the transformations in Table 1 except for the histories. Their choice of principal formula is free if $\text{Nx} = \perp$, but is restricted to the formula in Nx otherwise. If the decomposition χ of the principal $\langle\rangle$ -formula is a $\langle q \rangle$ -formula, we put Nx_1 of the child to be χ to enforce that χ is the principal formula of the child. The actions for uev ensure that $\text{uev}(\chi_1, \chi_2)$, where χ_1 is the principal $\langle\rangle$ -formula, inherits its value from the corresponding $\langle\rangle$ -formulae in the child: *e.g.* $\text{uev}(\langle\alpha; \beta\rangle\varphi, \chi_2) = \text{uev}_1(\langle\alpha\rangle\langle\beta\rangle\varphi, \chi_2)$ reverse-tracks the decomposition of $\langle\alpha; \beta\rangle\varphi$ into $\langle\alpha\rangle\langle\beta\rangle\varphi$. Also, $\text{uev}(\chi_1, \chi_2)$ is only defined if χ_1 is in the parent.

Universal Branching (β) Rules.

$$(\vee) \frac{(\varphi_1 \vee \varphi_2, \Gamma :: \text{Nx} :: \text{stat}, \text{uev})}{(\varphi_1, \Gamma :: \text{Nx} :: \text{stat}_1, \text{uev}_1) \mid (\varphi_2, \Gamma :: \text{Nx} :: \text{stat}_2, \text{uev}_2)} \quad \text{Nx} = \perp$$

$$([\?]) \frac{([\psi?]\varphi, \Gamma :: \text{Nx} :: \text{stat}, \text{uev})}{(\sim\psi, \Gamma :: \text{Nx} :: \text{stat}_1, \text{uev}_1) \mid (\varphi, \Gamma :: \text{Nx} :: \text{stat}_2, \text{uev}_2)} \quad \text{Nx} = \perp$$

Actions for $(\vee)$ and $([\?])$ for $i = 1, 2$:

$$\text{uev}'_i(\chi_1, \chi_2) := \begin{cases} \text{uev}_i(\chi_1, \chi_2) & \text{if } \chi_1 \in \Gamma \\ \perp & \text{otherwise} \end{cases}$$

$$(\langle \cup \rangle) \frac{(\langle \alpha_1 \cup \alpha_2 \rangle \varphi, \Gamma :: \text{Nx}, \text{BD} :: \text{stat}, \text{uev})}{(\langle \alpha_1 \rangle \varphi, \Gamma :: \text{Nx}_1, \text{BD}_1 :: \text{stat}_1, \text{uev}_1) \mid (\langle \alpha_2 \rangle \varphi, \Gamma :: \text{Nx}_2, \text{BD}_2 :: \text{stat}_2, \text{uev}_2)}$$

Side-condition for $(\langle \cup \rangle)$: $\text{Nx} \in \{\perp, \langle \alpha_1 \cup \alpha_2 \rangle \varphi\}$

Actions for $(\langle \cup \rangle)$ for $i = 1, 2$:

$$\begin{aligned} \text{Nx}_i &:= \text{Nxtst}(\langle \alpha_i \rangle \varphi) \\ \text{BD}_i &:= \text{BDtst}(\langle \alpha_i \rangle \varphi, \text{BD}) \\ \text{uev}'_i(\chi_1, \chi_2) &:= \begin{cases} \text{uev}_i(\langle \alpha_i \rangle \varphi, \chi_2) & \text{if } \chi_1 = \langle \alpha_1 \cup \alpha_2 \rangle \varphi \\ \text{uev}_i(\chi_1, \chi_2) & \text{if } \chi_1 \in \Gamma \\ \perp & \text{otherwise} \end{cases} \end{aligned}$$

$$(\langle * \rangle_1) \frac{(\langle \alpha * \rangle \varphi, \Gamma :: \text{Nx}, \text{BD} :: \text{stat}, \text{uev})}{(\varphi, \Gamma :: \text{Nx}_1, \text{BD}_1 :: \text{stat}_1, \text{uev}_1) \mid (\langle \alpha \rangle \langle \alpha * \rangle \varphi, \Gamma :: \text{Nx}_2, \text{BD}_2 :: \text{stat}_2, \text{uev}_2)}$$

Side-condition for $(\langle * \rangle_1)$: $\text{Nx} \in \{\perp, \langle \alpha * \rangle \varphi\}$ & $\langle \alpha * \rangle \varphi \notin \text{BD}$

Actions for $\langle * \rangle_1$:

$$\begin{aligned}
 \text{Nx}_1 &:= \text{Nxtst}(\varphi) \\
 \text{BD}_1 &:= \text{BDtst}(\varphi, \{\langle \alpha * \rangle \varphi\} \cup \text{BD}) \\
 \text{uev}'_1(\chi_1, \chi_2) &:= \begin{cases} \perp & \text{if } \chi_1 = \chi_2 = \langle \alpha * \rangle \varphi \\ \text{uev}_1(\varphi, \chi_2) & \text{if } \chi_1 = \langle \alpha * \rangle \varphi \neq \chi_2 \\ \text{uev}_1(\chi_1, \chi_2) & \text{if } \chi_1 \in \Gamma \\ \perp & \text{otherwise} \end{cases} \\
 \text{Nx}_2 &:= \text{Nxtst}(\langle \alpha \rangle \langle \alpha * \rangle \varphi) \\
 \text{BD}_2 &:= \text{BDtst}(\langle \alpha \rangle \langle \alpha * \rangle \varphi, \{\langle \alpha * \rangle \varphi\} \cup \text{BD}) \\
 \text{uev}'_2(\chi_1, \chi_2) &:= \begin{cases} \text{uev}_2(\langle \alpha \rangle \langle \alpha * \rangle \varphi, \chi_2) & \text{if } \chi_1 = \langle \alpha * \rangle \varphi \\ \text{uev}_2(\chi_1, \chi_2) & \text{if } \chi_1 \in \Gamma \\ \perp & \text{otherwise} \end{cases}
 \end{aligned}$$

The $\langle * \rangle_1$ -rule captures the fix-point nature of the $\langle * \rangle$ -formulae according to Prop. 2.6 as long as the principal formula is not blocked via BD. The choice of the principal formulae in the first child is either free if φ is not a $\langle q \rangle$ -formula or is φ if φ is a $\langle q \rangle$ -formula. In the latter case we also block the regeneration of $\langle \alpha * \rangle \varphi$ and thus avoid an “at a world” cycle by putting $\langle \alpha * \rangle \varphi$ into BD_1 . The right child is treated similarly but uses $\langle \alpha \rangle \langle \alpha * \rangle \varphi$ instead of φ .

Actions for all β -rules:

$$\begin{aligned}
 \text{stat} &:= \begin{cases} \mathbf{unsat} & \text{if } \text{stat}_1 = \mathbf{unsat} \ \& \ \text{stat}_2 = \mathbf{unsat} \\ \mathbf{open} & \text{if } \text{stat}_1 = \mathbf{open} \ \text{or} \ \text{stat}_2 = \mathbf{open} \\ \mathbf{barred} & \text{otherwise} \end{cases} \\
 \min_{\perp}(f, g)(\chi_1, \chi_2) &:= \begin{cases} \perp & \text{if } f(\chi_1, \chi_2) = \perp \ \text{or} \ g(\chi_1, \chi_2) = \perp \\ \min(f(\chi_1, \chi_2), g(\chi_1, \chi_2)) & \text{otherwise} \end{cases} \\
 \text{uev} &:= \begin{cases} \text{uev}_{\perp} & \text{if } \text{stat} \neq \mathbf{open} \\ \text{uev}'_1 & \text{if } \text{stat}_1 = \mathbf{open} \neq \text{stat}_2 \\ \text{uev}'_2 & \text{if } \text{stat}_1 \neq \mathbf{open} = \text{stat}_2 \\ \min_{\perp}(\text{uev}'_1, \text{uev}'_2) & \text{if } \text{stat}_1 = \mathbf{open} = \text{stat}_2 \end{cases}
 \end{aligned}$$

The intuitions are:

uev'_i : the definitions of uev'_i ensure that the pairs (χ_1, χ_2) , where χ_1 is the principal $\langle \rangle$ -formula, get the values from their corresponding $\langle \rangle$ -formulae in the children.

In the $\langle * \rangle_1$ -rule, a special case sets the value of $\text{uev}'_1(\chi_1, \chi_2)$ to $\perp$ if χ_1 and χ_2 are equal to the principal formula $\langle \alpha * \rangle \varphi$ of this rule since the eventuality $\langle \alpha * \rangle \varphi$ is no longer unfulfilled as the left child fulfils it. Note that $\text{uev}'(\chi_1, \chi_2)$ is only defined if χ_1 is in the parent.

$\min_\perp$: the definition of $\min_\perp$ ensures that we take the minimum of $f(\chi_1, \chi_2)$ and $g(\chi_1, \chi_2)$ only when both functions are defined for (χ_1, χ_2) .

uev : if $\text{stat} \neq \mathbf{open}$, the uev is irrelevant, so we arbitrarily set it as undefined. If only one child has $\text{stat} = \mathbf{open}$, we take its uev' . If both children have $\text{stat} = \mathbf{open}$, we take the minimum value of entries that are defined in uev'_1 and uev'_2 .

All previous rules modify existing uev -entries, but never create new ones. The next rule is the only rule that creates uev -entries (by identifying loops).

Existential Branching Rule.

$$\begin{array}{c}
 \langle a_1 \rangle \varphi_1, \dots, \langle a_n \rangle \varphi_n, \langle a_{n+1} \rangle \varphi_{n+1}, \dots, \langle a_{n+m} \rangle \varphi_{n+m}, [-] \Delta, \Gamma \\
 :: \text{HCr}, \text{Nx}, \text{BD}, \text{BB} :: \text{stat}, \text{uev} \\
 (\langle \rangle) \frac{}{\varphi_1, \Delta_1 :: \text{HCr}_1, \text{Nx}_1, \text{BD}_1, \text{BB}_1 \mid \dots \mid \varphi_n, \Delta_n :: \text{HCr}_n, \text{Nx}_n, \text{BD}_n, \text{BB}_n} \\
 :: \text{stat}_1, \text{uev}_1 \qquad \qquad \qquad :: \text{stat}_n, \text{uev}_n
 \end{array}$$

where:

- (1) $n + m \geq 0$
- (2) $\Gamma \subseteq (\text{AFml} \cup \{\neg q \mid q \in \text{AFml}\})$
- (3) $[-] \Delta \subseteq \{[a]\psi \mid a \in \text{APrg} \ \& \ \psi \in \text{Fml}\}$
- (4) $\Delta_i := \{\psi \mid [a_i]\psi \in [-] \Delta\}$ for $i = 1, \dots, n + m$
- (5) $\forall p \in \text{AFml}. \{p, \neg p\} \not\subseteq \Gamma$
- (6) $\forall i \in \{1, \dots, n\}. \forall j \in \{1, \dots, \text{len}(\text{HCr})\}. (\varphi_i, \{\varphi_i\} \cup \Delta_i) \neq \text{HCr}[j]$
- (7) $\forall k \in \{n + 1, \dots, n + m\}. \exists j \in \{1, \dots, \text{len}(\text{HCr})\}. (\varphi_k, \{\varphi_k\} \cup \Delta_k) = \text{HCr}[j]$

Actions for $(\langle \rangle)$: for $i = 1, \dots, n$: $\text{HCr}_i := \text{HCr} @ [(\varphi_i, \{\varphi_i\} \cup \Delta_i)]$,

$$\text{Nx}_i := \text{Nxtst}(\varphi_i), \quad \text{BD}_i := \emptyset, \quad \text{BB}_i := \emptyset$$

$$\text{stat} := \begin{cases} \mathbf{unsat} & \text{if } \exists i \in \{1, \dots, n\}. \text{stat}_i \neq \mathbf{open} \text{ or} \\ & (\exists \psi \in \text{Fml}(\ast). \varphi_i \in \text{pre}(\psi) \ \& \\ & \perp \neq \text{uev}_i(\varphi_i, \psi) > \text{len}(\text{HCr})) \\ \mathbf{open} & \text{otherwise} \end{cases}$$

$$\begin{aligned}
 \text{uev}_k(\cdot, \cdot) &:= j \in \{1, \dots, \text{len}(\text{HCr})\} \text{ such that } (\varphi_k, \{\varphi_k\} \cup \Delta_k) = \text{HCr}[j] \\
 &\text{for } k = n + 1, \dots, n + m
 \end{aligned}$$

$$\text{uev}(\chi_1, \chi_2) := \begin{cases} \text{uev}_i(\varphi_i, \chi_2) & \text{if stat} = \mathbf{open} \ \& \ \chi_2 \in \text{Fml}(\ast) \ \& \ \chi_1 \in \text{pre}(\chi_2) \\ & \& \ \chi_1 = \langle a_i \rangle \varphi_i \text{ for an } i \in \{1, \dots, n+m\} \\ \perp & \text{otherwise} \end{cases}$$

Some intuitions are in order:

- (1) If $n = 0$, the application of the rule generates no new nodes and stat vacuously evaluates to **open**. If $m = n = 0$, we additionally have $\text{uev} := \text{uev}_\perp$.
- (2) The set Γ contains only propositional atoms or their negations.
- (3) The set $[-]\Delta$ contains only formulae of the type $[a]\varphi$. Thus (2) and (3) imply that the $\langle \rangle$ -rule is applicable only if the node contains no α - or β -formulae.
- (4) The set Δ_i contains all formulae that must belong to the i^{th} child, which fulfils $\langle a_i \rangle \varphi_i$, so that we can build a Hintikka structure later on.
- (5) The node must not contain a contradiction.
- (6) 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 the formula set $\varphi_i \cup \Delta_i$ thereby generating the required successor for $\langle a_i \rangle \varphi_i$. Note that $\text{len}(\text{HCr})$ denotes the length of HCr.
- (7) If $m > 0$, then each $\langle a_k \rangle \varphi_k$ for $n+1 \leq k \leq n+m$ is “blocked” from creating its required child $\{\varphi_k\} \cup \Delta_k$ because some ancestor does the job. This ancestor must not only consist of the formulae $\{\varphi_k\} \cup \Delta_k$ but it must also have been created to fulfil $\langle a \rangle \varphi_k$ for some $a \in \text{APrg}$. Note that the values a_k and a are ignored when looking for loops since we are interested only in the contents of the required child.

HCr_i : is the HCr of the parent extended with an extra entry to record the “history” of worlds created on the path from the root down to the i^{th} child using “@” as list concatenation. Note that we store a *pair* $(\varphi_k, \varphi_k \cup \Delta_k)$, not just $\varphi_k \cup \Delta_k$. That is, we remember that the node $\varphi_k \cup \Delta_k$ was created to fulfil $\langle a \rangle \varphi_k$ for some $a \in \text{APrg}$.

stat: the parent is unsatisfiable if some child has stat $\neq \mathbf{open}$. But it is also unsatisfiable if some child, say the i^{th} , and some eventuality $\langle \alpha \ast \rangle \chi$ in it “loops lower” because $\varphi_i \in \text{pre}(\langle \alpha \ast \rangle \chi)$ and $\text{uev}_i(\varphi_i, \langle \alpha \ast \rangle \chi)$ is defined and greater than the length of the current HCr. Intuitively, the latter tells us that the eventuality $\langle \alpha \ast \rangle \chi$ occurs in the sub-tableau rooted at the parent but cannot be fulfilled.

uev_k : for $n+1 \leq k \leq n+m$, the k^{th} child is blocked by a higher (proxy) child. For every such k we set uev_k to be the *constant* function which maps every formula-pair to the *level* j of its proxy child. It is not too hard to see that uev_k is well-defined and unique. Note that the uev_k are just temporary functions used to define uev as explained next. The blocking child itself must have been created to fulfil a $\langle \rangle$ -formula $\langle a' \rangle \varphi_k$, as indicated by the first component of $\text{HCr}[j]$.

$\text{uev}(\chi_1, \chi_2)$: If stat = **unsat** then uev is undefined everywhere. Else, for each $\chi_1 = \langle a_i \rangle \varphi_i$ with $i \in \{1, \dots, n+m\}$, and each χ_2 with $\langle a_i \rangle \varphi_i \in \text{pre}(\chi_2)$, we take $\text{uev}(\langle a_i \rangle \varphi_i, \chi_2)$ from the formulae-pair (φ_i, χ_2) of the corresponding (real)

child if $\langle a_i \rangle \varphi_i$ is “unblocked”, or set it to the level of the proxy child higher in the branch if it is “blocked”. For all other formulae-pairs, uev is undefined. The intuition is that a defined uev(χ_1, χ_2) flags a “loop” which starts at the parent and eventually “loops” up to some blocking proxy. The value of uev(χ_1, χ_2) tells us the level of the proxy because we cannot classify this “loop” as “good” or “bad” until we backtrack to that level. The uev of each $\langle a_i \rangle \varphi_i$ is taken from *the* child created specifically to contain φ_i , a fact which is vital in the proofs.

BD_{*i*}, BB_{*i*}, Nx_{*i*}: each child has no blocked diamond- or box-formulae, and its principal formula is determined by the form of φ_i .

The $\langle \rangle$ - and *id*-rules are mutually exclusive via their side-conditions and the rules are designed so that at least one rule is applicable to any node.

4.2 A Tableau Procedure

As shown in the next section, we need to build only one fully expanded tableau, meaning that every node has exactly one rule applied to it. Hence, if multiple rules are applicable to a node, the choice of rule is immaterial, so any strategy for rule applications is fine. Of course, in our implementation, we give priority to the *id*-rule and the $\langle \rangle$ -rule, since they may close a branch sooner. Other heuristics, like preferring linear rules over branching rules, are also useful.

4.3 Termination, Soundness, and Completeness

Definition 4.4 Let $x = (\Gamma :: \text{HCr}, \text{Nx}, \text{BD}, \text{BB} :: \text{stat}, \text{uev})$ be a tableau node, φ a formula, and Δ a set of formulae. We write $\varphi \in x$ [$\Delta \subseteq x$] to mean $\varphi \in \Gamma$ [$\Delta \subseteq \Gamma$]. The parts of x are written as HCr_{*x*}, Nx_{*x*}, BD_{*x*}, BB_{*x*}, stat_{*x*}, and uev_{*x*}. Node x is *closed* iff stat_{*x*} = **unsat**, *open* iff stat_{*x*} = **open**, and *barred* iff stat_{*x*} = **barred**.

Definition 4.5 Let x be a $\langle \rangle$ -node in a tableau T (i.e. a $\langle \rangle$ -rule was applied to x). Then x is also called a *state* and the children of x are called *core-nodes*. Using the notation of the $\langle \rangle$ -rule, a formula $\langle a_i \rangle \varphi_i \in x$ is *blocked* iff $n + 1 \leq i \leq n + m$. For every not blocked $\langle a_i \rangle \varphi_i \in x$, the *successor* of $\langle a_i \rangle \varphi_i$ is the i^{th} child of the $\langle \rangle$ -rule. For every blocked $\langle a_i \rangle \varphi_i \in x$ there exists a unique core-node y on the path from the root of T to x such that $\{\varphi_i\} \cup \Delta_i$ is the set of formulae of y , and y is the successor of a formula $\langle a' \rangle \varphi_i$ in the parent of y . We call y the *virtual successor* of $\langle a_i \rangle \varphi_i$, and also call the formula φ_i in the (possibly virtual) successor of $\langle a_i \rangle \varphi_i$ a *core-formula*.

A state is just another term for a $\langle \rangle$ -node but a core-node can be any type of node (even a state). A state arises from a core-node by repeated applications of α - and β -rules. Note that the core-formula in a core-node y is well-defined and unique: if x_1 and x_2 are states and y is the (possibly virtual) successor of $\langle a_1 \rangle \varphi_1 \in x_1$ and $\langle a_2 \rangle \varphi_2 \in x_2$, then $\varphi_1 = \varphi_2$.

Let ϕ be a formula in negation normal form, and T an expanded tableau with root $r = (\{\phi\} :: [], \perp, \emptyset, \emptyset :: \text{stat}, \text{uev})$ with stat and uev determined by r ’s children.

Theorem 4.6 *T is a finite tree.*

Theorem 4.7 *If the root $r \in T$ is open, there is a Hintikka structure for ϕ .*

Theorem 4.8 *If the root $r \in T$ is not open then ϕ is not satisfiable.*

Theorem 4.9 *If $|\phi| = n$, our procedure has worst-case time complexity in $O(2^{2^n})$.*

Detailed proofs can be found in the extended version of this paper [2].

4.4 Two Fully Worked Examples

The first simple example illustrates how the procedure avoids infinite loops due to “at a world” cycles by blocking $\langle \alpha \rangle \varphi$ - and $[\alpha] \varphi$ -formulae from regenerating. The formula $\langle (q?) \rangle (p \wedge \neg p)$ is obviously not satisfiable. Hence, any expanded tableau with root $\langle (q?) \rangle (p \wedge \neg p)$ should not be open. Figure 1 shows such a tableau where each node is classified as a ρ -node if rule ρ is applied to that node in the tableau.

The initial formula $\langle (q?) \rangle (p \wedge \neg p)$ in node (1) is decomposed into a β_1 -child $p \wedge \neg p$ and a β_2 -child $\langle q? \rangle \langle (q?) \rangle (p \wedge \neg p)$ according to the $\langle * \rangle_1$ -rule. The formula $p \wedge \neg p$ in node (2) is then decomposed according to the $\wedge$ -rule and node (3) is marked as closed because it contains a contradiction. Node (2) inherits the status from node (3) unchanged according to the α -rules and, thus, is closed too.

Because the β_2 -formula $\langle q? \rangle \langle (q?) \rangle (p \wedge \neg p)$ is a $\langle \mathcal{A} \rangle$ -formula, the $\langle * \rangle_1$ -rule puts this formula into its Nx_2 , the Nx -value of node (4), and thus forces node (4) to have $\langle q? \rangle \langle (q?) \rangle (p \wedge \neg p)$ as its principal formula. For the same reason, the $\langle * \rangle_1$ -rule puts its own principal formula $\langle (q?) \rangle (p \wedge \neg p)$ into its BD_2 , the BD -value of node (4). Hence node (4) decomposes $\langle q? \rangle \langle (q?) \rangle (p \wedge \neg p)$ according to the $\langle ? \rangle$ -rule. Again, the resulting node (5) is forced to have $\langle (q?) \rangle (p \wedge \neg p)$ as its principal formula via its Nx -value, and gets its BD -value unchanged from node (4).

Node (5) has the same principal formula as node (1), so applying the $\langle * \rangle_1$ -rule to node (5) would cause the procedure to enter an “at a world” (infinite) cycle. Because the history BD of node (5) contains $\langle (q?) \rangle (p \wedge \neg p)$, the $\langle * \rangle_1$ -rule is blocked on node (5), but the $\langle * \rangle_2$ -rule is not. Hence the branch is terminated and the status of node (5) is set to **barred** (thereby avoiding the “at a world” cycle).

Node (4) inherits the status from node (5) unchanged and node (1) is marked **barred** also according to the definition of $stat$ in the β -rules. Therefore the tableau is not open. Note that the variable uev does not play a role in this example as it is irrelevant for nodes that are closed or barred.

The second example demonstrates the role of uev . The formula $[a^*]p \rightarrow [(a; a)^*]p$ is valid. Hence, its negation $\phi := [a^*]p \wedge \langle (a; a)^* \rangle \neg p$, which is already in negation normal form, is unsatisfiable and the root of any expanded tableau for ϕ should not be open. Figure 2 shows such a tableau. The unlabelled edges in Fig. 2 link states to core-nodes. We omit the histories BD and BB as they do not play an important role in this example. Each partial function UEV_i maps the formula-pair (ψ_i, χ_i) in Table 2 to 1 and is undefined otherwise as explained below. The histories are $H_1 := [(\varphi_1, \Delta_1)]$ where $\varphi_1 := \langle a \rangle \langle (a; a)^* \rangle \neg p$ and $\Delta_1 := \{[a^*]p, \langle a \rangle \langle (a; a)^* \rangle \neg p\}$ and $H_2 := H_1 @ [(\varphi_2, \Delta_2)]$ where $\varphi_2 := \langle (a; a)^* \rangle \neg p$ and $\Delta_2 := \{[a^*]p, \langle (a; a)^* \rangle \neg p\}$.

Fig. 1. A first example: a closed tableau for $\langle (q?)^* \rangle (p \wedge \neg p)$

The dotted frame at (7a) indicates that its child, an *id*-node, is not shown due to space restrictions. Thus the marking of the nodes (3a) and (7a) in Fig. 2 with **unsat** is straightforward. The leaf (9) is a $\langle \rangle$ -node, but it is “blocked” from creating its successor containing $\Delta := \{[a^*]p, \langle a \rangle \langle (a; a)^* \rangle \neg p\}$ because there is a $j \in \mathbb{N}$ such that $\text{HCr}_9[j] = H_2[j] = \langle \langle a \rangle \langle (a; a)^* \rangle \neg p, \Delta \rangle$: namely $j = 1$. Thus the $\langle \rangle$ -rule computes $\text{UEV}_1(\langle a \rangle \varphi_1, \langle (a; a)^* \rangle \neg p) = 1$ as stated above and also puts $\text{stat}_9 := \text{open}$. As node (7a) is closed, nodes (8), (7b), (7), (6), and (5) inherit their functions UEV_i from their open children via the corresponding α - and β -rules.

The crux of our method occurs at node (4), a $\langle \rangle$ -node with $\text{HCr}_4 = \square$ and hence $\text{len}(\text{HCr}_4) = 0$. The $\langle \rangle$ -rule thus finds a child node (5) and a pair of formulae $(\psi, \chi) := (\langle a \rangle \langle (a; a)^* \rangle \neg p, \langle (a; a)^* \rangle \neg p)$ where ψ is a core-formula, $\psi \in \text{pre}(\chi)$, and $1 = \text{UEV}_4(\psi, \chi) = \text{uev}_5(\psi, \chi) > \text{len}(\text{HCr}_4) = 0$. Thus node (4) “sees” a child (5) that “loops lower”, meaning that node (5) is the root of an “isolated” subtree which fails to fulfil its eventuality $\langle (a; a)^* \rangle \neg p$. The $\langle \rangle$ -rule marks (4) as closed via $\text{stat}_4 = \text{unsat}$. The propagation of **unsat** to the root is simple.

What if the omitted child of (7a), and hence (7a) itself, had been open? Then UEV_3 in (7) would be undefined everywhere via the $\langle * \rangle_1$ -rule, regardless of uev_{7b} . Thus $\langle (a; a)^* \rangle \neg p$ in (7) would be fulfilled via the β_1 -child (7a). Hence UEV_4 would be undefined everywhere, and node (4) would not be closed.

Fig. 2. A second example: a closed tableau for $[a^*]p \wedge \langle (a; a)^* \rangle \neg p$

5 Conclusion and Further Work

We have given a sound and complete tableau algorithm for checking *PDL*-satisfiability with 2EXPTIME worst-case time-complexity rather than EXPTIME. The following further practical and theoretical work remain. First, we believe that a small refinement of our histories will allow our calculus to classify a loop as “bad” or “good” at the looping leaf, as is done by Baader’s procedure [4], but with no pre-computation of automata. Thus it should be possible to extend DLP to handle our method. Further experimental work is required to determine if such an extension will remain practical. Second, recent work has shown that global caching can indeed deliver optimality of tableau procedures soundly [11]. The histories used in our calculus make it harder to extend sound global caching to it since nodes are now sensitive to their context in the tree under construction. Further theoretical work is required to extend sound global caching to handle such context sensitivity.

References

- [1] Abate, P., R. Goré, and F. Widmann. One-pass tableaux for computation tree logic. In N. Dershowitz and A. Voronkov, editors, *Proc. LPAR 2007*, LNCS 4790:32–46, Springer, 2007.
- [2] Abate, P., R. Goré, and F. Widmann. An On-the-fly Tableau-based Decision Procedure for PDL-Satisfiability. arXiv:0711.1016 (<http://arxiv.org/abs/0711.1016>).
- [3] Alberucci L., and G. Jäger. About cut elimination for logics of common knowledge. *Annal of Pure and Applied Logic*, 133(1-3):73–99, 2005.
- [4] Baader, F. Augmenting concept languages by transitive closure of roles: an alternative to terminological cycles. Technical Report RR-90-13, DFKI, Kaiserslautern, Germany, 1990.
- [5] Ben-Ari, M., Z. Manna, and A. Pnueli. The temporal logic of branching time. In *Proc. Eighth ACM Symposium on Principles of Programming Languages*, pages 164–176, 1981.
- [6] Bhat, G., and R. Cleaveland. Efficient on-the-fly model checking for CTL*. In *Proc. Symposium on Logics in Computer Science*, pages 388–397, 1995.
- [7] Cleaveland, R. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica*, 27:725–747, 1990.
- [8] Emerson, E. A., and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Science*, 30:1–24, 1985.
- [9] Fischer, M. J., and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer Systems and Science*, 18:194–211, 1979.
- [10] Giacomo, G. D., and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, 160:109–169, 2000.
- [11] Goré, R., and L. A. Nguyen. Exptime tableaux for ALC using sound global caching. In *DL’07: Proc. International Workshop on Description Logics, 2007*, 8 pages.
- [12] Horrocks, I., and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, OUP, 1999.

Table 2
Definitions for the example in Fig. 2

UEV_i	$i = 1$	$i = 2$	$i = 3$	$i = 4$
ψ_i	$\langle a \rangle \langle a \rangle \langle (a; a)^* \rangle \neg p$	$\langle a; a \rangle \langle (a; a)^* \rangle \neg p$	$\langle (a; a)^* \rangle \neg p$	$\langle a \rangle \langle (a; a)^* \rangle \neg p$
χ_i	$\langle (a; a)^* \rangle \neg p$	$\langle (a; a)^* \rangle \neg p$	$\langle (a; a)^* \rangle \neg p$	$\langle (a; a)^* \rangle \neg p$

- [13] Jäger, G., M. Kretz, and T. Studer. Cut-free common knowledge. *Journal of Applied Logic*, to appear.
- [14] Pratt, V. Semantical considerations on Floyd-Hoare logic. In *Proc. 17th IEEE Symposium on Foundations Of Computer Science*, pages 109–121, 1976.
- [15] Pratt, V. A near-optimal method for reasoning about action. *Journal of Computer Systems and Science*, 20:231–254, 1980.
- [16] Schmidt, R., and D. Tishkovsky. Personal communication, September 2007. <http://www.cs.man.ac.uk/~schmidt/pdl-tableau>
- [17] Schwendimann, S. A new one-pass tableau calculus for PLTL. In H. de Swart, editor, *Proc. TABLEAUX'98*, LNAI 1397:277–291. Springer, 1998.
- [18] Wolper, P. Temporal logic can be more expressive. *Information and Computation*, 56:72–99, 1983.