A Cut-Free ExpTime Tableau Decision Procedure for the Logic Extending Converse-PDL with Regular Inclusion Axioms*

Linh Anh Nguyen
Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

Abstract. We give the first cut-free ExpTime (optimal) tableau decision procedure for the logic CPDL_{reg}, which extends Converse-PDL with regular inclusion axioms characterized by finite automata. The logic CPDL_{reg} is the combination of Converse-PDL and regular grammar logic with converse. Our tableau decision procedure uses global state caching and has been designed to increase efficiency and allow various optimization techniques, including on-the-fly propagation of local and global (in)consistency.

1 Introduction

In this paper we study automated reasoning in the modal logic CPDL_{reg}, which is a combination of CPDL (propositional dynamic logic with converse [16,2,21]) and REG^c (regular grammar logic with converse [5,22]). The logic CPDL is widely used in many areas, including program verification, theory of action and change as well as knowledge representation (see, e.g., [16,9,7]). The logic REG^c can be used for modeling and reasoning about epistemic states of multi-agent systems [12] and Web ontologies [22,17].

In the papers [6,8] written jointly with Dunin-Keplicz and Szalas, we show that CPDL_{reg} is a logical formalism suitable for expressing complex properties of agents’ cooperation in terms of beliefs, goals and intentions. The logic CPDL_{reg} can also be used as a description logic (DL). As a combination of CPDL and REG^c, it allows to use role constructors (by program constructors of CPDL) like the DL CIQ [9] and to use role inclusion axioms (by inclusion axioms of REG^c) similarly to the DL SROIQ [17] (but by using automata instead of grammar rules).

The mentioned works [6,8] present a tableau calculus leading to the first ExpTime (optimal) tableau decision procedure for CPDL_{reg}. Observing the rules of that calculus, it can be seen that the general satisfiability checking problem in CPDL_{reg} is reducible to the satisfiability checking problem in CPDL. However, translating the finite automata of a given CPDL_{reg} logic L into regular expressions and then applying an ExpTime (optimal) decision procedure like the ones given by us and Szalas [21] or by Goré and Widmann [15] for the resulted satisfiability problem in CPDL may require double exponential time in the size of the original problem. The reason is that the regular expressions resulted from translating the finite automata specifying L may have exponential lengths. One can also consider the method of translating REG^c into CPDL given in [4] by Demri and de Nivelle. However, as CPDL_{reg} is much more complicated than REG^c, it is not trivial at all how to generate that method for translating CPDL_{reg} into CPDL and how efficient the approach would be. Therefore, as stated in [5], it is worth studying the direct approach for automated reasoning in CPDL_{reg}.

The tableau calculus given in our joint papers [6,8] for CPDL_{reg} uses analytic cuts and therefore is not efficient in practice. In this paper we improve that calculus by eliminating cuts to give the first cut-free ExpTime (optimal) tableau decision procedure for CPDL_{reg}. Our calculus uses global state caching as in the works [14,15] by Goré and Widmann. It also uses local caching for non-states of tableaux.

The idea of global caching comes from Pratt’s paper on PDL [24]. In [11,12,13] together with Goré we formalized and applied it to the modal logics REG (regular grammar logics),

* This work is supported by Polish National Science Centre grant 2011/01/B/ST6/02759.
BReg (regular modal logics of agent beliefs) and the description logic $SHI$ to obtain $\text{ExpTime}$ (optimal) tableau decision procedures for these logics. Later, together with Szabas we extended the method to give $\text{ExpTime}$ (optimal) tableau decision procedures for the logics PDL \cite{23}, CPDL \cite{21} and $\text{REG}^c$ \cite{22}. In \cite{13,21,22} we used analytic cuts to deal with inverse roles and converse modal operators. As cuts are not efficient in practice, Goré and Widmann developed the first cut-free $\text{ExpTime}$ tableau decision procedures, based on global state caching, for the DL $\text{ALC}T$ \cite{14} and CPDL \cite{15}.

In the current paper, we use the idea of global state caching and a slightly different technique to deal with converse modal operators for CPDL$_{reg}$. We check “compatibility” of a non-state $w$ with its predecessor-state $v$ as soon as possible and do not require $w$ to be “saturated”. In \cite{14,15} Goré and Widmann require $w$ to be “saturated” before checking the compatibility, which has a serious drawback, for example, when the label of $v$ is of the form $\{p, \langle \sigma \rangle [\sigma^{-}] \neg p, \sigma[(p_1 \lor q_1) \land \ldots \land (p_n \lor q_n)]\}$. Besides, we tend to give a higher priority for the current branch even when it involves converse, while Goré and Widmann \cite{14,15} tend to delay solving incompatibility w.r.t. converse.\footnote{We use the verb “tend” because any systematic search strategy is applicable for both \cite{14,15} and the current paper. Nevertheless, the “tendency” can still be observed.} Our technique of dealing with converse modal operators for CPDL$_{reg}$ can be described as follows: if a state $v$ is “too small” due to the lack of a formula $\varphi$ and a non-state $u$ is the unique predecessor of $v$ then delete the edge $(u, v)$ and connect $u$ to two nodes $v_1$ and $v_2$ (which are created when necessary) such that $v_1$ extends $v$ with $\varphi$ and $v_2$ is similar to $v$ but always “disallows” $\varphi$ (i.e. if any non-state $w$ having $v_2$ as the predecessor-state requires adding $\varphi$ to $v_2$ then we give $w$ status $\text{unsat}$). For simplicity, in the current version of this paper we do not consider separately the case when $v$ is “too small” due to the lack of a set $\{\varphi_1, \ldots, \varphi_k\}$ of formulas (like an “alternative set” \cite{14,15}) with $k > 1$. The slowdown is not high as we check “compatibility” between nodes as soon as possible. A generalization of our method for this case is straightforward: we delete the edge $(u, v)$ and connect $u$ to nodes $v_0, v_1, \ldots, v_k$ (which are created when necessary) such that $v_0$ extends $v$ with $\{\varphi_1, \ldots, \varphi_k\}$ and, for each $1 \leq i \leq k$, $v_i$ is similar to $v$ but always disallows $\varphi_i$.

Apart from \cite{6,8,14,15}, some other papers that are most related to ours are: Pratt’s paper \cite{24} on PDL, Demri’s paper \cite{3} of REG, the paper \cite{5} by Demri and de Nivelle on REG\textsuperscript{c}, the paper \cite{2} by De Giacomo and Massacci on CPDL. Also, one can translate the satisfiability problem of CPDL$_{reg}$ into the satisfiability problem of modal $\mu$-calculus with converse or the emptiness problem of automata on infinite trees. The former one is often translated into the latter one (see Vardi’s work \cite{26}). Checking emptiness of an automaton on infinite trees can be reduced to checking who has a winning strategy in a graph of game, which is similar to “and-or” graphs used in the current paper. However, the construction of such a tree automaton itself is a weak point of the approach. As stated in \cite{11}, optimization techniques have not been adequately studied for theorem proving based on tree automata, and this is why the approach is not used in practice for theorem proving in description logics (which are closely related to modal logics).

The rest of this paper is structured as follows. In Section 2 we give definitions for the logic CPDL$_{reg}$. In Section 3 we present our tableau calculus for CPDL$_{reg}$. In Section 5 we give an $\text{ExpTime}$ tableau decision procedure for checking satisfiability of a set of formulas w.r.t. a set of global assumption in CPDL$_{reg}$. We conclude in Section 7.

### 2 Preliminaries

Let $\Sigma_+$ be a finite set of symbols. For $\sigma \in \Sigma_+$, we use $\sigma^-$ to denote a fresh symbol, called the converse of $\sigma$. Let $\Sigma^- = \{\sigma^- \mid \sigma \in \Sigma_+\}$ and $\Sigma = \Sigma_+ \cup \Sigma^-$. We call $\Sigma$ an alphabet with converse. For $\varphi = \sigma^- \in \Sigma_-$, define $\varphi^- = \sigma$.

A context-free semi-Thue system $S$ over $\Sigma$ is a finite set of context-free production rules over alphabet $\Sigma$. We say that $S$ is symmetric if, for every rule $\sigma \rightarrow q_1 \ldots q_k$ of $S$, the rule...

...
programs also belongs to \( S \). A context-free semi-Thue system \( S \) over \( \Sigma \) is called a regular semi-Thue system \( S \) over \( \Sigma \) if, for every \( \sigma \in \Sigma \), the set of words derivable from \( \sigma \) using the system is a regular language over \( \Sigma \).

A context-free semi-Thue system is like a context-free grammar, but it has no designated start symbol and there is no distinction between terminal and non-terminal symbols.

Similarly to [5], we assume that any regular semi-Thue system \( S \) is given together with a mapping \( A \) that associates each \( \sigma \in \Sigma \) with a finite automaton \( A_\sigma \) recognizing words derivable from \( \sigma \) using \( S \). We assume that for \( \sigma \in \Sigma \), the word \( \sigma \) is derivable from \( \sigma \) by such a system. We call \( A \) the mapping specifying the finite automata of \( S \).

A finite automaton \( A \) over alphabet \( \Sigma \) is a tuple \( \langle \Sigma, Q, I, \delta, F \rangle \), where \( Q \) is a finite set of states, \( I \subseteq Q \) is the set of initial states, \( \delta \subseteq Q \times \Sigma \times Q \) is the transition relation, and \( F \subseteq Q \) is the set of accepting states. A run of \( A \) on a word \( q_0 \ldots q_k \) is a finite sequence of states \( q_0, q_1, \ldots, q_k \) such that \( q_0 \in I \) and \( \delta(q_{i-1}, q_i, q_i) \) holds for every \( 1 \leq i \leq k \). It is an accepting run if \( q_k \in F \). We say that \( A \) accepts a word \( w \) if there exists an accepting run of \( A \) on \( w \). The set of words accepted by \( A \) is denoted by \( L(A) \).

We use \( \mathcal{PROP} \) to denote the set of propositions (i.e., atomic formulas) and use letters like \( p \) to denote its elements. We call elements of \( \Sigma^+ \) atomic programs, and call elements of \( \Sigma \) simple programs. We denote simple programs by letters like \( \sigma \) and \( \varphi \).

Formulas and programs in the base language of \( \text{CPDL}_{\text{reg}} \) are defined respectively by the following BNF grammar rules:

\[
\varphi ::= \top | \bot | p | \neg \varphi | \varphi \rightarrow \varphi | \varphi \land \varphi | \varphi \lor \varphi | (\alpha)\varphi | [\alpha]\varphi
\]

\[
\alpha ::= \sigma | \alpha \land \alpha | \alpha \lor \alpha | \alpha^* | \alpha^- | \varphi^2
\]

We use letters like \( \alpha, \beta \) to denote programs, and \( \varphi, \psi, \xi \) to denote formulas. Given binary relations \( R_1 \) and \( R_2 \), by \( R_1 \circ R_2 \) we denote their relational composition.

A Kripke model is a pair \( M = \langle \Delta^M, .^M \rangle \), where \( \Delta^M \) is a set of states, and \( .^M \) is an interpretation function that maps each proposition \( p \) to a subset \( p^M \) of \( \Delta^M \), and each atomic program \( \sigma \in \Sigma^+ \) to a binary relation \( \sigma^M \) on \( \Delta^M \). The interpretation function is extended to interpret complex formulas and complex programs as follows:

\[
\top^M = \Delta^M, \quad \bot^M = \emptyset
\]

\[
(\neg \varphi)^M = \Delta^M \setminus \varphi^M, \quad (\varphi \rightarrow \psi)^M = (\neg \varphi \lor \psi)^M
\]

\[
(\varphi \land \psi)^M = \varphi^M \cap \psi^M, \quad (\varphi \lor \psi)^M = \varphi^M \cup \psi^M
\]

\[
([\alpha]\varphi)^M = \{ x \in \Delta^M | \exists y[\alpha^M(x, y) \land \varphi^M(y)] \}
\]

\[
([\alpha]\varphi)^M = \{ x \in \Delta^M | \forall y[\alpha^M(x, y) \rightarrow \varphi^M(y)] \}
\]

\[
(\alpha \land \beta)^M = \alpha^M \cup \beta^M, \quad (\alpha^* M) = (\alpha^M)^* \]

\[
(\alpha^-)^M = (\alpha^M)^{-1} = \{ (y, x) | (x, y) \in \alpha^M \}
\]

\[
(\varphi^2)^M = \{ (x, x) | \varphi^M(x) \}
\]

We write \( M, w \models \varphi \) to denote \( w \in \varphi^M \). For a set \( X \) of formulas, we write \( M, w \models X \) to denote that \( M, w \models \varphi \) for all \( \varphi \in X \). If \( M, w \models \varphi \) (respectively, \( M, w \models X \)), then we say that \( M \) satisfies \( \varphi \) (respectively, \( X \)) at \( w \), and that \( \varphi \) (respectively, \( X \)) is satisfied at \( w \) in \( M \). We say that \( M \) validates \( X \) (and \( X \) is valid in \( M \)) if \( M, w \models X \) for all \( w \in \Delta^M \).

Note that the definition of \( (\sigma^-)^M \) is compatible with the assumption \( (\sigma^-)^{-1} = \sigma \).

Let \( S \) be a symmetric regular semi-Thue system over \( \Sigma \). The \( \text{CPDL}_{\text{reg}} \) logic corresponding to \( S \), denoted by \( \text{CPDL}(S) \), is characterized by the class of Kripke models \( M \) such that, for

\[\text{Note that it is undecidable to check whether a context-free semi-Thue system is regular since it is undecidable whether the language generated by a linear grammar is regular [15].}\]
every rule \( \sigma \rightarrow \sigma_1 \ldots \sigma_k \) of \( S \), we have that \( \sigma_1^M \circ \ldots \circ \sigma_k^M \subseteq \sigma^M \). Such a structure is called an L-model, for \( L = \text{CPDL}(S) \).

Let \( L \) be a CPDL\(_\text{reg} \) logic and \( X, \Gamma \) be finite sets of formulas. We say that \( X \) is \( L \)-satisfiable w.r.t. the set \( \Gamma \) of global assumptions if there exists an \( L \)-model that validates \( \Gamma \) and satisfies \( X \) at some state.

We say that two sets \( X \) and \( Y \) of formulas are \( L \)-equivalent if for every \( L \)-model \( M \) and every state \( w \) of \( M \), \( (M, w \models X) \) iff \( (M, w \models Y) \).

A formula or a program is in \( \text{negation-and-converse normal form} \) (NCNF) if it does not use the connective \( \rightarrow \), uses the operator \( \neg \) only immediately before propositions, and uses the converse program constructor \( \neg \) only for atomic programs.

Every formula \( \varphi \) (respectively, program \( \alpha \)) can be transformed to a formula \( \varphi' \) (respectively, program \( \alpha' \)) in NCNF that is equivalent to \( \varphi \) (respectively, \( \alpha \)) in the sense that for every Kripke model \( M \), \( \varphi^M = (\varphi')^M \) (respectively, \( \alpha^M = (\alpha')^M \)). For example, the NCNF of formula
\[ \neg[(\sigma_1 \cup \sigma_2); \sigma_3^\gamma; (\neg \neg p)^\gamma][q \vee \neg r] \] is \( (\neg p)^\gamma; (\sigma_3^\gamma^*) \cup (\sigma_1^\gamma \cup \sigma_2^\gamma)(\neg q \land r) \). In this paper we assume that formulas and programs are represented in NCNF and write \( \overline{\varphi} \) to denote the NCNF of \( \neg \varphi \).

The alphabet \( \Sigma(\alpha) \) of a program \( \alpha \) and the \( \text{regular language} \ L(\alpha) \) generated by \( \alpha \) are specified as follows:

\[ \Sigma(\sigma) = \Sigma, \quad L(\sigma) = \{ \sigma \} \]
\[ \Sigma(\varphi?) = \{ \varphi? \}, \quad L(\varphi?) = \{ \varphi? \} \]
\[ \Sigma(\beta; \gamma) = \Sigma(\beta) \cup \Sigma(\gamma), \quad L(\beta; \gamma) = L(\beta) \cup L(\gamma) \]
\[ \Sigma(\beta \cup \gamma) = \Sigma(\beta) \cup \Sigma(\gamma), \quad L(\beta \cup \gamma) = L(\beta) \cup L(\gamma) \]
\[ \Sigma(\beta^*) = \Sigma(\beta), \quad L(\beta^*) = (L(\beta))^* \]

where for sets of words \( M \) and \( N \), \( M \cdot N = \{ \alpha \beta \mid \alpha \in M, \beta \in N \} \), \( M^0 = \{ \varepsilon \} \) (where \( \varepsilon \) denotes the empty word), \( M^{n+1} = M \cdot M^n \) for \( n \geq 0 \), and \( M^* = \bigcup_{n \geq 0} M^n \).

We will use letters like \( \omega \) to denote either a simple program from \( \Sigma \) or a test (of the form \( \varphi? \)). A word \( \omega_1 \ldots \omega_k \in L(\alpha) \) can be treated as the program \( (\omega_1; \ldots ; \omega_k) \), especially when it is interpreted in a Kripke model. As a finite automaton \( A \) over alphabet \( \Sigma(\alpha) \) corresponds to a program (the regular expression recognizing the same language), it is interpreted in a Kripke model as follows:

\[ A^M = \bigcup \{ \gamma^M \mid \gamma \in L(A) \}. \] (1)

### 3 A Tableau Calculus for CPDL\(_{\text{reg}} \)

From now on, let \( S \) be a symmetric regular semi-Thue system over \( \Sigma \), \( A \) be the mapping specifying the finite automata of \( S \), and \( L \) be the CPDL\(_{\text{reg}} \) logic corresponding to \( S \). In this section we present a tableau calculus for checking \( L \)-satisfiability.

For each program \( \alpha \), let \( \mathcal{A}_\alpha \) be a finite automaton recognizing the regular language \( L(\alpha) \). For each program \( \alpha \notin \Sigma \), let \( \mathcal{A}_\alpha \) be a finite automaton recognizing the language \( L(\alpha') \), where \( \alpha' \) is obtained from \( \alpha \) by substituting each \( \sigma \in \Sigma \) not inside any test by a regular expression representing \( L(\mathcal{A}_\sigma) \).

The automaton \( \mathcal{A}_\alpha \) can be constructed from \( \alpha \) in polynomial time, and \( \mathcal{A}_\alpha \) can be constructed in polynomial time in the length of \( \alpha \) and the sizes of the automata \( (\mathcal{A}_\sigma)_{\sigma \in \Sigma} \). Roughly speaking, \( \mathcal{A}_\alpha \) can be obtained from \( \mathcal{A}_\alpha \) by simultaneously substituting each transition \( (q_1, \sigma, q_2) \) by the automaton \( \mathcal{A}_\sigma \).

From now on, let \( X \) and \( \Gamma \) be finite sets of formulas in NCNF of the base language. For the tableau calculus defined here for checking \( L \)-satisfiability of \( X \) w.r.t. the set \( \Gamma \) of global assumptions we extend the base language with the auxiliary modal operators \( \Box_\sigma, [A, q] \) and \( (A, q) \), where \( \sigma \in \Sigma, A \) is either \( \mathcal{A}_\alpha \) or \( \mathcal{A}_\alpha \) for some program \( \alpha \) occurring in \( X \) or \( \Gamma \) in the form \([\alpha] \varphi \) or \((\alpha) \varphi \), and \( q \) is a state of \( A \). Here, \([A, q] \) and \((A, q) \) stand respectively for \([[A, q]] \)}

\(^3\) Note that \( \Sigma(\alpha) \) contains not only simple programs but also expressions of the form \((\varphi?)\), and a program \( \alpha \) is a regular expression over its alphabet \( \Sigma(\alpha) \).
and \( \langle (A,q) \rangle \), where \((A,q)\) is the automaton that differs from \(A\) only in that \(q\) is its only initial state. We call \([A,q]\) (respectively, \(\langle A,q \rangle \)) a **universal** (respectively, **existential** automaton-modal operator).

In the **extended language**, if \( \varphi \) is a formula, then \( \Box_\sigma \varphi \), \([A,q] \varphi \) and \( \langle A,q \rangle \varphi \) are also formulas. A formula \( \Box_\sigma \varphi \) has the same semantics as \( [\sigma] \varphi \), i.e. \( (\Box_\sigma \varphi)^M = ([\sigma] \varphi)^M \). The semantics of formulas \([A,q] \varphi \) and \( \langle A,q \rangle \varphi \) are defined as usual, treating \((A,q)\) as a program with semantics specified by \( I \).

Despite that \( \Box_\sigma \varphi \) has the same semantics as \( [\sigma] \varphi \), the operator \( \Box_\sigma \) behaves differently from \( [\sigma] \) in our calculus. Given a Kripke model \( M \) and a state \( w \in \Delta^M \), we have that \( w \in ([A,q] \varphi)^M \) (respectively, \( w \in ((A,q) \varphi)^M \)) if

\[
w_k \in \varphi^M \text{ for all (respectively, some) } w_k \in \Delta^M \text{ such that there exist a word } \omega_1 \ldots \omega_k
\]

(with \( k \geq 0 \)) accepted by \((A,q)\) with \((w,w_k) \in (\omega_1; \ldots; \omega_k)^M \).

The condition \((w,w_k) \in (\omega_1; \ldots; \omega_k)^M \) means there exist states \( w_0 = w, w_1, \ldots, w_{k-1} \) of \( M \) such that, for each \( 1 \leq i \leq k \), if \( w_i \in \Sigma \) then \((w_{i-1}, w_i) \in \omega_i^M \), else \( w_i = (\psi_i) \) for some \( \psi_i \) and \( w_{i-1} = w_i \) and \( w_i \in \psi_i^M \). Clearly, \( \langle A,q \rangle \) is dual to \([A,q] \) in the sense that \( \langle A,q \rangle \varphi \equiv \neg[A,q] \neg \varphi \) for any formula \( \varphi \).

We will use the following convention:

- given a finite automaton \( A \), we always assume that \( A = \langle \Sigma_A, Q_A, I_A, \delta_A, F_A \rangle \)
- for \( q \in Q_A \), we define \( \delta_A(q) = \{ (\omega, q') \mid (q, \omega, q') \in \delta_A \} \).

**Lemma 3.1.** Let \( M \) be a Kripke model, \( \alpha \) be a program, \( \varphi \) be a formula, and \( A \) be a finite automaton over \( \Sigma(\alpha) \). Then:

1. if \( I_A = \{ q_1, \ldots, q_k \} \) then \( A^M = (A, q_1)^M \cup \ldots \cup (A, q_k)^M \)
2. \( A^M = K^M \)
3. if \( I_{A_\alpha} = \{ q_1, \ldots, q_k \} \) then \( (\alpha \varphi)^M = ([A_\alpha, q_1] \varphi \lor \ldots \lor [A_\alpha, q_k] \varphi)^M \)
4. if \( M \) is an \( L \)-model then
   a. \( K^M = A^M \)
   b. if \( I_{A_\alpha} = \{ q_1, \ldots, q_k \} \) then \( ([\alpha] \varphi)^M = ([A_\alpha, q_1] \varphi \land \ldots \land [A_\alpha, q_k] \varphi)^M \).

**Proof.** The assertions 1 and 2 clearly hold. The assertion 3 follows from the assertions 1 and 2. For the assertion 4a, just note that \( M \) is an \( L \)-model and, for every \( \sigma \in \Sigma \), \( A_\sigma \) accepts \( \sigma \). The assertion 4b follows from the assertions 1, 2, and 4a.

In what follows we define tableaux as rooted “and-or” graphs. Such a graph is a tuple \( G = (V,E,\nu) \), where \( V \) is a set of nodes, \( E \subseteq V \times V \) is a set of edges, \( \nu \in V \) is the root, and each node \( v \in V \) has a number of attributes. If there is an edge \((v,w) \in E \) then we call \( v \) a **predecessor** of \( w \), and call \( w \) a **successor** of \( v \). The set of all attributes of \( v \) is called the **contents** of \( v \). Attributes of tableau nodes are:

- \( \text{Type}(v) \in \{ \text{state}, \text{non-state} \} \). If \( \text{Type}(v) = \text{state} \) then we call \( v \) a **state**, else we call \( v \) a **non-state** (or an **internal** node). A state is never directly connected to a state.
- \( \text{Status}(v) \in \{ \text{unexpanded}, \text{expanded}, \text{incomplete}, \text{unsat}, \text{sat} \} \).
- \( \text{Label}(v) \) is a finite set of formulas, called the label of \( v \).
- \( \text{RFmls}(v) \) is a finite set of formulas, called the set of reduced formulas of \( v \).
- \( \text{DFmls}(v) \) is a finite set of formulas, called the set of disallowed formulas of \( v \).
- \( \text{StatePred}(v) \in V \cup \{ \text{null} \} \) is called the state-predecessor of \( v \). It is available only when \( \text{Type}(v) = \text{non-state} \). If \( v \) is a non-state and \( G \) has no paths connecting a state to \( v \) then \( \text{StatePred}(v) = \text{null} \). Otherwise, \( G \) has exactly one state \( u \) that is connected to \( v \) via a path not containing any other states. In that case, \( \text{StatePred}(v) = u \).
CELabel(v) is a formula of the form \(\langle \sigma \rangle \varphi\). It is called the coming edge label of \(v\) and is available only when \(v\) is a successor of a state. Informally, if \(\text{CELabel}(v) = \langle \sigma \rangle \varphi\) and \(u\) is the predecessor of \(v\) then \(\langle \sigma \rangle \varphi \in \text{Label}(u)\) and this formula is realized at \(u\) by creating a transition to \(v\) (transitions are done only for states).

\(\text{ATPred}(v) \in V\) is called the after-transition-predecessor of \(v\). It is available only when \(\text{Type}(v) = \text{non-state}\). If \(v\) is a non-state and \(v_0 = \text{StatePred}(v) \neq \text{null}\) then there is exactly one successor \(v_1\) of \(v_0\) such that every path connecting \(v_0\) to \(v\) must go through \(v_1\), and we have that \(\text{ATPred}(v) = v_1\). If \(v\) is a non-state and \(\text{StatePred}(v) = \text{null}\) then \(\text{ATPred}(v)\) is set to the root \(\nu\).

\(\text{FmlSC}(v)\) is a formula called the formula suggested by converse. It is available only when \(\text{Type}(v) = \text{state}\) and \(\text{Status}(v) = \text{incomplete}\). In that case, it means that some node \(w\) with \(\text{StatePred}(w) = v\) wants \(\varphi \in \text{Label}(v) \cup \text{RFmls}(v)\) but it does not hold. The formula \(\varphi\) is expected to be present at \(v\) when, for \(v_1 = \text{ATPred}(w)\), \(\text{CELabel}(v_1)\) is of the form \(\langle \sigma \rangle \psi\) and \(\Box_{\sigma \varphi} \in \text{Label}(w)\).

We define

\[
\begin{align*}
\text{AFmls}(v) &= \text{Label}(v) \cup \text{RFmls}(v) \\
\text{NDFmls}(v) &= \left\{ \varphi \mid \varphi \in \text{DFmls}(v) \right\} \\
\text{FullLabel}(v) &= \text{AFmls}(v) \cup \text{NDFmls}(v) \\
\text{Kind}(v) &= \begin{cases}
\text{and-node} & \text{if } \text{Type}(v) = \text{state} \\
\text{or-node} & \text{if } \text{Type}(v) = \text{non-state}
\end{cases}
\end{align*}
\]

\(\text{BeforeFormingState}(v) = v\) has a successor which is a state

\(\text{AfterTrans}(v) = (\text{ATPred}(v) = v)\)

\(\text{AFmls}(v)\) is called the available formulas of \(v\). In an “and-or” graph, states play the role of “and”-nodes, while non-states play the role of “or”-nodes.\(^4\)

By the local graph of a state \(v\) we mean the subgraph of \(G\) consisting of all the paths starting from \(v\) and not containing any other states. Similarly, by the local graph of a non-state \(v\) we mean the subgraph of \(G\) consisting of all the path starting from \(v\) and not containing any states.

We apply global state caching in the sense that if \(v_1\) and \(v_2\) are different states then \(\text{Label}(v_1) \neq \text{Label}(v_2)\) or \(\text{RFmls}(v_1) \neq \text{RFmls}(v_2)\) or \(\text{DFmls}(v_1) \neq \text{DFmls}(v_2)\). If \(v\) is a non-state such that \(\text{AfterTrans}(v)\) then we also apply global caching for the local graph of \(v\). That is, if \(w_1\) and \(w_2\) are different nodes of the local graph of \(v\) then \(\text{Label}(w_1) \neq \text{Label}(w_2)\) or \(\text{RFmls}(w_1) \neq \text{RFmls}(w_2)\) or \(\text{DFmls}(w_1) \neq \text{DFmls}(w_2)\).

Our calculus \(CL\) for the CPDL\(_{reg}\) logic \(L\) will be specified, amongst others, by a finite set of tableau rules, which are used to expand nodes of tableaux. A tableau rule is specified with the following information:

- the kind of the rule: an “and”-rule or an “or”-rule
- the conditions for applicability of the rule (if any)
- the priority of the rule
- the number of successors of a node resulting from applying the rule to it, and the way to compute their contents.

Tableau rules are usually written downwards, with a set of formulas above the line as the premise, which represents the label of the node to which the rule is applied, and a number of sets of formulas below the line as the (possible) conclusions, which represent the labels of the successor nodes resulting from the application of the rule. Possible conclusions of an “or”-rule are separated by \(|\), while conclusions of an “and”-rule are separated by \(\&\). If a rule is a unary rule (i.e. a rule with only one possible conclusion) or an “and”-rule then its conclusions are

\(^4\) But this is not necessarily true for (“and-or” graphs in) other logics.
“firm” and we ignore the word “possible”. The meaning of an “or”-rule is that if the premise is $L$-satisfiable w.r.t. $\Gamma$ then some of the possible conclusions are also $L$-satisfiable w.r.t. $\Gamma$, while the meaning of an “and”-rule is that if the premise is $L$-satisfiable w.r.t. $\Gamma$ then all of the conclusions are also $L$-satisfiable w.r.t. $\Gamma$ (possibly in different states of the model under construction).

Such a representation gives only a part of the specification of the rules.

We use $Y$ to denote a set of formulas, write $Y, \varphi$ to denote $Y \cup \{ \varphi \}$ and write $Y, \Gamma$ to denote $Y \cup \Gamma$. Our tableau calculus $\mathcal{CL}$ for the CPDL$_{reg}$ logic $L$ w.r.t. the set $\Gamma$ of global assumptions consists of the rules which are partially specified in Table 1 together with two special rules ($\text{forming-state}$) and ($\text{conv}$), which will be explained later.

For any rule of $\mathcal{CL}$ except ($\text{forming-state}$) and ($\text{conv}$), the distinguished formulas of the premise are called the principal formulas of the rule. The rules ($\text{forming-state}$) and ($\text{conv}$) have no principal formulas.

As usual, we assume that, for each rule of $\mathcal{CL}$ described in Table 1 the principal formulas are not members of the set $Y$ which appears in the premise of the rule.

The rule (trans) is the only “and”-rule and the only transitional rule. Instantiating this rule, for example, to the set $\{ \langle \sigma \rangle p, \langle \sigma \rangle q, \sqcap \sigma r \}$ as the premise and $\Gamma = \{ x \}$ we get two conclusions: $\{ p, r, s \}$ and $\{ q, r, s \}$. Expanding a state $v$ in a tableau by the rule (trans), each successor $w_j$ of $v$ is created due to a corresponding principal formula $\langle \sigma_j \rangle \varphi_i$ of the rule and we have $\text{CELabel}(w_i) = \langle \sigma_i \rangle \varphi_i$.

The other rules of $\mathcal{CL}$ are “or”-rules, which are also called static rules. The intuition behind distinguishing between static/transitional is that static rules do not change the state of the model under construction, while each conclusion of the transitional rule forces a move to a new state. The transitional rule is used to expand states of tableaux, while the static rules are used to expand non-states of tableaux.

For any state $w$, every predecessor $v$ of $w$ is always a non-state. Such a node $v$ is expanded and connected to $w$ by the static rule (forming-state). The nodes $v$ and $w$ correspond to the same state of the Kripke model under construction. In other words, the rule (forming-state) “transforms” a non-state to a state. The idea is to separate internal nodes (i.e. non-states) from states, which are globally cached. The rule (forming-state) guarantees that, if $\text{BeforeFormingState}(v)$ holds then $v$ has exactly one successor, which is a state.

Expanding a non-state $v$ of a tableau by a static rule $\rho \in \{(\land), (\lor), (\text{aut}_2), (\text{aut}_1), ([A]), ([A]_f), (\sqcap \tau)\}$ which uses $\varphi$ as the principal formula we put $\varphi$ into the set $RFmls(w)$ of each successor $w$ of $v$ by setting $RFmls(w) := RFmls(v) \cup \{ \varphi \}$. We use $RFmls(w)$ to disallow expanding $w$ by static rules which use a formula from $RFmls(w)$ as the principal formula (i.e. to block reducing the formulas from $RFmls(w)$ twice). If a non-state $v$ is expanded by a static rule $\rho \in \{ ([A]), ([A]_f), (\sqcap \tau), (\text{conv}) \}$ and $w$ is a successor of $v$ then we set $RFmls(w) := RFmls(v)$. Thus, we do not use the attribute $RFmls$ to disallow the rules $\langle A \rangle, ([A]_f), (\sqcap \tau)$ (in order to be able to fulfill eventualities of the form $A, q/\varphi$). If $v$ is expanded by the rule (trans) and $w$ is a successor of $v$ then we set $RFmls(w) := \emptyset$.

Let $v_0, v_1, \ldots, v_k$ be a path of a tableau such that $k \geq 1$, $Type(v_0) = \text{state}$ and $Type(v_i) = \text{non-state}$ for $1 \leq i \leq k$. Suppose that $\text{CELabel}(v_1) = \langle \sigma \rangle \psi$ and $\sqcap \sigma \neg \varphi \in Label(v_k)$. If $v_0$ corresponds to a state $x$ of a Kripke model $M$, then all $v_1, \ldots, v_k$ correspond to a state $y$ of $M$ such that $(x, y) \in \sigma^M$. The formulas from $Label(v_k)$ are supposed to be satisfied at $y$ in $M$, which causes $\varphi$ satisfied at $x$ in $M$. So, $\varphi$ is expected to belong to $AFmls(v_0)$ (i.e. we should realize $\varphi$ at $v_0$). What should be done in the case $\varphi \notin AFmls(v_0)$? We will not simply add $\varphi$ to $Label(v_0)$, as $v_k$ is only one of possibly many “or”-descendants of $v_0$, and adding $\varphi$ to $Label(v_0)$ may affect the other “or”-descendants of $v_0$ (which is not allowed). If $\varphi \in DFmls(v_0)$, which means $\varphi$ is disallowed in $v_0$, then $\text{Status}(v_k)$ becomes $\text{unsat}$, which intuitively means that the “combination” of $v_0$ and $v_k$ is $L$-unsatisfiable w.r.t. $\Gamma$. In the other case, $\text{Status}(v_0)$ becomes $\text{incomplete}$, the predecessors of $v_0$ will be re-expanded by the rule (conv) either to have $\varphi$ (by
adding \( \varphi \) to the attribute \textit{Label} or to disallow \( \varphi \) (by adding \( \varphi \) to the attribute \textit{DFmls}). For details, see Steps 3-6 of procedure \textbf{Apply} given on page 11 (where \( w \) plays the role of \( v_0 \)).

The priorities of the tableau rules are as follows (the bigger, the stronger): unary static rules except (\textit{forming-state}): 5; non-unary static rules: 4; (\textit{forming-state}): 3; (\textit{trans}): 2; (\textit{conv}): 1\(^5\)

The conditions for applying a rule \( \rho \neq (\textit{conv}) \) to a node \( v \) are as follows:

- the rule has \textit{Label}(\( v \)) as the premise
- all the conditions accompanying with \( \rho \) in Table 1 are satisfied
- if \( \rho = (\textit{trans}) \) then \textit{Type}(\( v \)) = \textit{state}
- if \( \rho \neq (\textit{trans}) \) then \textit{Type}(\( v \)) = \textit{non-state} and
  - if \( \rho \in \{ (\wedge), (\lor), (\textit{aut}_{\wedge}), (\textit{aut}_{\lor}), ([A]), ([A]_f), (\Box_f) \} \) then the principal formula of \( \rho \) does not belong to \textit{RFmls}(\( v \))
  - no static rule with a higher priority is applicable to \( v \).

Application of a tableau rule \( \rho \) to a node \( v \) is specified by procedure \textbf{Apply}(\( \rho, v \)) given on page 11. Auxiliary functions are defined on page 9. Procedures used for updating and propagating statuses of nodes are defined on page 11. The main function \textbf{Tableau}(\( X, \Gamma \)) is also defined on page 11. It returns a rooted “and-or” graph called a \textit{CL-tableau} for (\( X, \Gamma \)).

\(^5\) The rule (\textit{conv}) is not chosen like the others but only invoked at another part of the algorithm (namely Procedure \textbf{UpdateStatus}).
Example 3.2. Consider the regular grammar logic with converse $L$ that corresponds to the following semi-Thue system over alphabet $\{\sigma, q, \sigma^-, q^-\}$:

$$\{q \to \sigma q, q^- \to \sigma, \sigma^- \to q^-\}.$$ 

The set of words derivable from $q$ is characterized by $(\sigma^-)^*(\sigma + q)$. Let

$$A_q = \langle\{\sigma, q, \sigma^-, q^-\}, \{0, 1\}, \{0\}, \{(0, \sigma^-), (0, \sigma, 1), (0, q, 1)\}, \{1\}\rangle.$$ 

In Figures 1 and 2 we give an “and-or” graph for $\langle\sigma(p \land \neg [q])\rangle$, $\emptyset$ w.r.t. $CL$. The nodes are numbered when created and are expanded using DFS (depth-first search). At the end the root receives status unsat. Therefore, by Theorem 3.3 $\langle\sigma(p \land \neg [q])\rangle$ is $L$-unsatisfiable.

Observe that:

- An application of the rule (conv) to a node $v$ may cause a sequence of other applications of this rule to “ancestor nodes” of $v$ and may put formulas far back of the tableau.
Lemma 3.3. Let $h = |\Sigma|$ and let $k$ be the sum of the sizes of the automata $A_\sigma$ (for $\sigma \in \Sigma$), $l$ be the size of $X \cup \Gamma$, and $n$ be the size of $\text{closure}_L(X \cup \Gamma)$. Let $G = (V, E, \nu)$ be a CL-tableau for $(X, \Gamma)$. Then $n$ is polynomial in $k, h, l$ and, for every $v \in V$:

1. The sets $\text{Label}(v)$, $\text{RFmls}(v)$, $\text{DFmls}(v)$ and $\text{NDFmls}(v)$ contain only formulas from $\text{closure}_L(X \cup \Gamma)$.

2. $\text{RFmls}(v)$ does not contain formulas of the form $\langle A, q \rangle \varphi$ or $\langle \psi ? \rangle \varphi$. 

- If the logic $L$ is essentially a $\text{REG}^\dagger$ logic, then no formulas of the form $(A, q) \varphi$ occur in tableaux (and we do not have to check “global consistency”).

Define the length of a formula $\varphi$ to be the number of symbols occurring in $\varphi$. For example, the length of $(A, q) \psi$ is the length of $\psi$ plus 5, treating $A_\sigma$ as a symbol. Define the size of a finite set of formulas to be the length of the conjunction of its formulas. Define the size of a finite automaton $(\Sigma, Q, I, \delta, F)$ to be $|Q| + |I| + |\delta| + |F|$.

For a set $Y$ of formulas, the set of basic subformulas of $Y$, denoted by $\text{bsf}(Y)$, consists of all formulas $\varphi$ and $\overline{\varphi}$ of the base language such that either $\varphi \in Y$ or $\varphi$ is a subformula of some formula of $Y$. The set $\text{closure}_L(Y)$ is defined to be the union of $\text{bsf}(Y)$ and the following two sets:

- $\{\langle A_\alpha, q \rangle \varphi, \sqcap_\omega \langle A_\alpha, q \rangle \varphi | (\alpha) \varphi \in \text{bsf}(Y), q \in Q_{A_\alpha} \text{ and } \omega \in \Sigma_{A_\alpha} \}$
- $\{\langle h_{A_\alpha}, q \rangle \varphi, \langle \omega \rangle \langle h_{A_\alpha}, q \rangle \varphi | (\alpha) \varphi \in \text{bsf}(Y), \alpha \notin \Sigma, \alpha \text{ is not a test, } q \in Q_{h_{A_\alpha}} \text{ and } \omega \in \Sigma_{h_{A_\alpha}} \}$
Function Tableau($X, \Gamma$)

**Input:** finite sets $X$ and $\Gamma$ of concepts in NCNF of the base language.

**Global data:** a rooted graph $(V, E, \nu)$.

1. $\nu := \text{NewSucc}(\text{null, non-state, null, } X \cup \Gamma, \emptyset, \emptyset)$;
2. if $\text{Unsat}(\nu)$ then $\text{Status}(\nu) := \text{unsat}$;
3. else if $\text{Sat}(\nu)$ then $\text{Status}(\nu) := \text{sat}$;
4. while ($\nu := \text{ToExpand()}$) $\neq \text{null}$ do
   5. choose a tableau rule $p$ different from (conv) and applicable to $\nu$;
   6. $\text{Apply}(p, \nu); //$ defined on page 10
   7. foreach $w \in V$ such that $\text{Status}(w) =$ incomplete do
      8. delete the local graph of $w$ except the node $w$
5. return $(V, E, \nu)$

Procedure UpdateStatus($v$)

**Global data:** a rooted graph $(V, E, \nu)$.

**Input:** a node $v \in V$ with $\text{Status}(v) =$ expanded.

1. if $\text{Kind}(v) =$ or-node then
   2. if some successors of $v$ have status sat then $\text{Status}(v) := \text{sat}$
   3. else if all successors of $v$ have status unsat then $\text{Status}(v) := \text{unsat}$
   4. else if a successor of $v$ has status incomplete then $\text{Apply}((\text{conv}), v)$
5. else // $\text{Kind}(v) =$ and-node
   6. if all successors of $v$ have status sat then $\text{Status}(v) := \text{sat}$
   7. else if some successors of $v$ have status unsat then $\text{Status}(v) := \text{unsat}$

Procedure PropagateStatus($v$)

**Global data:** a rooted graph $(V, E, \nu)$.

**Input:** a node $v \in V$ with $\text{Status}(v) \in \{\text{incomplete, unsat, sat}\}$.

1. foreach predecessor $u$ of $v$ with $\text{Status}(u) =$ expanded do
   2. $\text{UpdateStatus}(u)$;
   3. if $\text{Status}(u) \in \{\text{incomplete, unsat, sat}\}$ then $\text{PropagateStatus}(u)$

3. If $v$ is a state then $\text{Label}(v) \cap \text{RFmls}(v) = \emptyset$ and $\text{Label}(v)$ does not contain formulas of the form $\varphi \land v$, $\varphi \lor v$, $[a, q] \varphi$, $\Box(w) \varphi$, $\langle A, q \rangle \varphi$ or $\langle \beta \rangle \varphi$ with $\beta \notin \Sigma$.
4. (a) $\text{Label}(v)$ is $L$-equivalent to $\text{Label}(v) \cup \text{RFmls}(v)$
   (b) $\text{Label}(v) \setminus \text{RFmls}(v)$ is $L$-equivalent to $\text{Label}(v) \cup \text{RFmls}(v)$.

Furthermore, the graph $G$ has no more than $2^{O(n)}$ nodes and can be constructed in $2^{O(n)}$ steps.

**Proof.** Clearly, $n$ is polynomial in $k.h.l$. The assertions [13] should be clear. The assertion [4] can be proved by induction in a straightforward way. Since global state caching is used, by assertion [1] $G$ has no more than $2^{O(n)}$ states. Similarly, since global caching is used for the local graphs of non-states, the local graph of each non-state has no more than $2^{O(n)}$ nodes. Hence, $G$ has no more than $2^{O(n)}$ nodes. Each node of the graph may be re-expanded at most once, using the rule (conv). Expansion of a node can be done in polynomial time in the size of the graph. Hence the graph can be constructed in $2^{O(n)}$ steps.

A marking of a $\text{CL}$-tableau $G$ is a subgraph $G_m$ of $G$ such that:

- the root of $G$ is the root of $G_m$
- if $v$ is a node of $G_m$ and is an “or”-node of $G$ then at least one edge $(v, w)$ of $G$ is an edge of $G_m$
- if $v$ is a node of $G_m$ and is an “and”-node of $G$ then every edge $(v, w)$ of $G$ is an edge of $G_m$
- if $(v, w)$ is an edge of $G_m$ then $v$ and $w$ are nodes of $G_m$.
Fig. 1. An illustration for Example 3.2 part I. The graph (a) is the “and-or” graph constructed until checking “compatibility” of the node (6) w.r.t. to the node (1). In each node, we display the name of the rule expanding the node and the formulas of the label of the node. The attribute $DFmls$ of each of the displayed nodes is an empty set. The node (2) is the only state. As an example, we have $StatePred((6)) = (2), ATPred((6)) = (3)$ and $CELabel((3)) = ⟨σ⟩(p ∧ [σ]¬p)$. Since $□p ∩ [A, 0] ¬p ∈ Label((6))$, $Status((2))$ is set to incomplete and $FmlSC((2))$ is set to $[A, 0]¬p$. This results in the graph (b). The construction is then continued by applying the rule $(conv)$ to (1) and deleting the nodes (3)-(6). See Figure 2 for the continuation.
Fig. 2. An illustration for Example 3.2 part II. This is a fully expanded “and-or” graph for $(\{\langle \sigma \rangle (p \land [\rho] \neg p)\}, \emptyset)$ w.r.t. $\text{CL}$. The node (1) is re-expanded by the rule (conv). As in the part I, in each node we display the name of the rule expanding the node and the formulas of the label of the node. We display also the attribute $DFmls$ of the nodes (8) and (9). This attribute of any other node is an empty set. The nodes (2), (10) and (14) are the only states. The node (13) receives status $\text{unsat}$ because $(p, \neg p) \subseteq \text{Label}((13))$. After that the nodes (12)-(7) receive status $\text{unsat}$ in subsequent steps. The node (18) receives status $\text{unsat}$ because $[A, 0] \neg p \in DFmls((14))$. After that the nodes (17)-(14), (8), (1) receive status $\text{unsat}$ in subsequent steps.
Let $G$ be a $CL$-tableau for $(X, \Gamma)$, $G_m$ be a marking of $G$, $v$ be a node of $G_m$, and $\langle A, q \rangle \varphi$ be a formula of $Label(v)$. A trace of $\langle A, q \rangle \varphi$ in $G_m$ starting from $v$ is a sequence $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ such that:

- $v_0 = v$ and $\varphi_0 = \langle A, q \rangle \varphi$
- for every $1 \leq i \leq k$, $(v_{i-1}, v_i)$ is an edge of $G_m$
- for every $1 \leq i \leq k$, $\varphi_i$ is a formula of $Label(v_i)$ such that
  - if $\varphi_{i-1}$ is not a principal formula of the tableau rule expanding $v_{i-1}$ then the rule must be a static rule and $\varphi_i = \varphi_{i-1}$
  - else if the rule is $((\exists A))$ or $((\exists A)')$ then $\varphi_{i-1}$ is of the form $\langle A, q' \rangle \varphi$ and $\varphi_i$ is the formula obtained from $\varphi_{i-1}$
  - else if the rule is $((\land ?))$ then $\varphi_{i-1}$ is of the form $\langle \psi, A, q' \rangle \varphi$ and $\varphi_i = \langle A, q' \rangle \varphi$
  - else the rule is $(\text{trans})$, $\varphi_{i-1}$ is of the form $\langle \sigma, A, q' \rangle \varphi$ and is the coming edge label of $v_i$, and $\varphi_i = \langle A, q' \rangle \varphi$

A trace $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ of $\langle A, q \rangle \varphi$ in a marking $G_m$ is called a $\Diamond$-realization in $G_m$ for $\langle A, q \rangle \varphi$ at $v_0$ if $\varphi_k = \varphi$.

A marking $G_m$ of a $CL$-tableau $G$ is consistent if:

- **local consistency:** $G_m$ does not contain nodes with status $\text{unsat}$; and
- **global consistency:** for every node $v$ of $G_m$, every formula $\langle A, q \rangle \varphi$ of $Label(v)$ has a $\Diamond$-realization (starting from $v$) in $G_m$.

**Theorem 3.4 (Soundness and Completeness).** Let $S$ be a symmetric regular semi-Thue system over $\Sigma$, $A$ be the mapping specifying the finite automata of $S$, and $L$ be the $\text{CPDL}_{reg}$ logic corresponding to $S$. Let $X$ and $\Gamma$ be finite sets of formulas in $\text{NCNF}$ of the base language, and $G$ be a $CL$-tableau for $(X, \Gamma)$. Then $X$ is $L$-satisfiable w.r.t. the set $\Gamma$ of global assumptions iff $G$ has a consistent marking.

### 4 Proofs of Soundness and Completeness

Let $G$ be a $CL$-tableau for $(X, \Gamma)$. For each $v \in V$ with $\text{Status}(v) \in \{\text{incomplete}, \text{unsat}, \text{sat}\}$, let $\text{DSTimeStamp}(v)$ be the step number at which $\text{Status}(v)$ is changed to its final value (i.e. determined to be $\text{incomplete}$, $\text{unsat}$, or $\text{sat}$). $\text{DSTimeStamp}$ stands for “determined-status timestamp”.

#### 4.1 Soundness

**Lemma 4.1.** Let $G = (V, E, v)$ be a $CL$-tableau for $(X, \Gamma)$. Then no node with status $\text{incomplete}$ is reachable from $v$.

**Proof.** By Steps 4, 25, 26 of procedure $\text{Apply}$ and Step 4 of procedure $\text{UpdateStatus}$. <

**Lemma 4.2.** If a non-state $v$ has $w_1, \ldots, w_k$ as all the successors then, for every $L$-model $M$ and every $x \in \Delta^M$, we have that $M, x \models \text{FullLabel}(v)$ iff there exists $1 \leq i \leq k$ such that $M, x \models \text{FullLabel}(w_i)$. <

The proof of this lemma is straightforward.

**Lemma 4.3.** Let $G = (V, E, v)$ be a $CL$-tableau for $(X, \Gamma)$. Let $v \in V$ be a node with $\text{Status}(v) = \text{unsat}$. Then:

- case $\text{Type}(v) = \text{state}$ : $\text{FullLabel}(v)$ is $L$-unsatisfiable w.r.t. $\Gamma$  
- case $\text{Type}(v) = \text{non-state}$ and $\text{StatePred}(v) = \text{null}$ : $\text{FullLabel}(v)$ is $L$-unsatisfiable w.r.t. $\Gamma$
case Type(v) = non-state and v₀ = StatePred(v) ≠ null: there are no L-model M and x, y ∈ ΔM such that M validates Γ, (x, y) ∈ σM, M, x ⊨ FullLabel(v₀) and M, y ⊨ FullLabel(v), where v₁ = ATPred(v) and CELabel(v₁) is of the form ⟨σ⟩ψ.

Proof. We prove this by induction on DSTimeStamp(v).

If ⊨ Label(v) or there exists {φ, ψ} ⊆ Label(v) then FullLabel(v) is clearly L-unsatisfiable w.r.t. Γ. So, in the rest of this proof, we exclude this case.

Consider the case when Type(v) = state.

We have that Kind(v) = and-node. There exists a successor w of v with Status(w) = unsat and DSTimeStamp(w) < DSTimeStamp(v). By the inductive assumption, FullLabel(w) is L-unsatisfiable w.r.t. Γ. Since DFmls(w) = ∅, by assertion 4 of Lemma 3.3 it follows that Label(w) is L-unsatisfiable w.r.t. Γ. Hence Label(v) and FullLabel(v) are L-unsatisfiable w.r.t. Γ.

Consider the case when Type(v) = non-state and StatePred(v) = null.

We have that Kind(v) = or-node. Let w₁, ..., wₖ be all the successors of v. We must have that, for all 1 ≤ i ≤ k, Status(wᵢ) = unsat and DSTimeStamp(wᵢ) < DSTimeStamp(v). By the inductive assumption, for all 1 ≤ i ≤ k, FullLabel(wᵢ) is L-unsatisfiable w.r.t. Γ. By Lemma 4.2 FullLabel(v) is L-unsatisfiable w.r.t. Γ.

Consider the case when Type(v) = non-state and v₀ = StatePred(v) ≠ null.

– Case when v has a successor w being a state: The node v must be expanded by the rule (forming-state). Since w is the only successor of v, it must be that Status(w) = unsat and DSTimeStamp(w) < DSTimeStamp(v). By the inductive assumption, FullLabel(w) is L-unsatisfiable w.r.t. Γ. Hence FullLabel(v) = FullLabel(w) is also L-unsatisfiable w.r.t. Γ.

– Case when v has a successor and all the successors w₁, ..., wₖ of v are non-states.
  • If Status(v) was set to unsat by Step 20 of procedure Apply (with w = v) then, for v₁ = ATPred(v), CELabel(v₁) is of the form ⟨σ⟩ψ, Label(v) contains a formula of the form □σ¬φ and φ $\notin$ AFmls(v₀), φ ∈ DFmls(v₀). For the sake of contradiction, suppose that there exist an L-model M and x, y ∈ ΔM such that M validates Γ, (x, y) ∈ σM, M, x ⊨ FullLabel(v₀) and M, y ⊨ FullLabel(v). Since □σ¬φ ∈ Label(v), it follows that M, y ⊨ □σ¬φ, and hence M, x ⊨ φ. This contradicts the facts that φ ∈ DFmls(v₀) and M, x ⊨ FullLabel(v₀).

  • Consider the remaining case. It must be that, for all 1 ≤ i ≤ k, Status(wᵢ) = unsat and DSTimeStamp(wᵢ) < DSTimeStamp(v). By the inductive assumption, for all 1 ≤ i ≤ k, FullLabel(wᵢ) is L-unsatisfiable w.r.t. Γ. By Lemma 4.2 FullLabel(v) is L-unsatisfiable w.r.t. Γ.

Lemma 4.4 (Soundness). Let G = (V, E, ν) be a CL-tableau for (X, Γ). Suppose that X is L-satisfiable w.r.t. the set Γ of global assumptions. Then G has a consistent marking.

Proof. Since Type(ν) = non-state, Label(ν) = X ∪ Γ and DFmls(ν) = ∅, we have that FullLabel(ν) is L-satisfiable w.r.t. Γ. Let M be an L-model validating Γ and let τ be a state of M such that M, τ ⊨ FullLabel(ν).

Observe that, for any v ∈ V and any x ∈ ΔM, if M, x ⊨ FullLabel(v) then:

– if v is a non-state then v has a successor w such that M, x ⊨ FullLabel(w) (by Lemma 4.2)

– if v is a state then, for every successor w of v with CELabel(w) of the form ⟨σ⟩ψ, there exists y ∈ ΔM such that (x, y) ∈ σM and M, y ⊨ FullLabel(w).

Therefore, starting from ν ∈ V and τ ∈ ΔM it is straightforward to construct a marking G_m of G together with a map g that associates each edge (ν, w) of G_m with a pair (x, y) ∈ ΔM × ΔM such that:

– if v is a non-state then x = y

Note that CELabel(w) ∈ Label(v).
– if $v$ is a state and $CELabel(w)$ is of the form $\langle \sigma, \psi \rangle$ then $(x, y) \in \sigma^M$
– $M, x \models FullLabel(v)$ and $M, y \models FullLabel(w)$.

By Lemma 4.3 for every node $v$ of $G_m$, $Status(v) \neq unsat$. Therefore, $G_m$ satisfies the local consistency property. We now show that $G_m$ satisfies the global consistency property. Let $v_0$ be a node of $G_m$ and let $(A, q) \varphi \in Label(v_0)$. We show that $(A, q) \varphi$ has a $\Diamond$-realization in $G_m$ starting from $v_0$. By the map $g$, there exists $u \in \Delta^M$ such that $M, u \models FullLabel(v_0)$. Thus $M, u \models (A, q) \varphi$ and therefore,

there exist a word $\gamma = \omega_1 \ldots \omega_k \in \mathcal{L}(\langle A, q \rangle)$ with $k \geq 0$, an accepting run $q_0 = q, q_1, \ldots, q_k$ of $(A, q)$ on $\gamma$, and states $u_0 = u, u_1, \ldots, u_k$ of $M$ such that $(A, q, \varphi)$ and, for $1 \leq i \leq k$, if $\omega_i \in \Sigma$ then $(u_{i-1}, u_i) \in \omega_i^M$, else $\omega_i$ is of the form $(\psi_i ?)$ and $u_{i-1} = u_i \in \psi_i^M$.

We construct a $\Diamond$-realization $(v_0, \varphi_0), \ldots, (v_h, \varphi_h)$ in $G_m$ for $(A, q) \varphi$ at $v_0$ and a map $f : \{0, \ldots, h\} \rightarrow \{0, \ldots, k\}$ such that $f(0) = 0$, $f(h) = k$, and for every $0 \leq i < h$, if $f(i) = j$ then $f(i + 1)$ is either $j$ or $j + 1$. We maintain the following invariants for $0 \leq i \leq h$:

- the sequence $(v_0, \varphi_0), \ldots, (v_i, \varphi_i)$ is a trace of $(A, q) \varphi$ in $G_m$ (3)
- $FullLabel(v_i)$ is satisfied at the state $u_{f(i)}$ of $M$ (4)
- if $f(i) = j < k$ then $\varphi_i = (A, q_j) \varphi$ or $\varphi_i = \langle \omega_{j+1} \rangle(A, q_{j+1}) \varphi$ (5)
- if $f(i) = k$ then $\varphi_i = (A, q_k) \varphi$ or $\varphi_i = \varphi$ (6)

With $\varphi_0 = (A, q_0) \varphi$ and $f(0) = 0$, the invariants clearly hold for $i = 0$.

Set $i := 0$. While $\varphi_i \neq \varphi$ do:

1. Set $j := f(i)$.
2. Case $v_i$ is expanded using a static rule $\rho$ and $\varphi_i$ is the principal formula:
   (a) Case $j < k$ : Since $\rho$ is a static rule, by the invariant (5), we must have $\varphi_i = (A, q_j) \varphi$ or $\varphi_i = \langle \omega_{j+1} \rangle(A, q_{j+1}) \varphi$ with $\omega_{j+1} = (\psi_{j+1} ?)$.
   Consider the case $\varphi_i = (A, q_j) \varphi$. The applied rule $\rho$ is thus either $(\langle A \rangle)$ or $(\langle A \rangle f)$. Let $\varphi_{i+1} = \langle \omega_{j+1} \rangle(A, q_{j+1}) \varphi$ and let $v_{i+1}$ be the successor of $v_i$ with $\varphi_{i+1}$ replacing $\varphi_i$. By (2), $\varphi_{i+1}$ is satisfied at $u_j$ in $M$, and hence, by the invariant (1), $FullLabel(v_{i+1})$ is satisfied at $u_j$ in $M$. Let $f(i + 1) = j$ and set $i := i + 1$. Clearly, the invariants still hold.
   Now consider the case $\varphi_i = \langle \omega_{j+1} \rangle(A, q_{j+1}) \varphi$ with $\omega_{j+1} = (\psi_{j+1} ?)$. The applied rule $\rho$ is thus $(\Diamond \varphi)$. Let $\varphi_{i+1} = \langle A, q_{j+1} \rangle \varphi$ and let $v_{i+1}$ be the only successor of $v_i$. By (2), both $\varphi_{i+1}$ and $\langle A, q_{j+1} \rangle \varphi$ are satisfied at $u_j$ in $M$, and hence, by the invariant (1), $FullLabel(v_{i+1})$ is satisfied at $u_j = u_{j+1}$ in $M$. Let $f(i + 1) = j + 1$ and set $i := i + 1$. Clearly, the invariants still hold.
3. Case $v_i$ is expanded using a static rule $\rho$ and either $\rho$ does not have principal formulas or $\varphi_i$ is not the principal formula:
   – Case $\rho$ does not have principal formulas or the principal formula is not of the form $(A', q') \varphi'$: Let $\varphi_{i+1} = \varphi_i$ and $f(i + 1) = f(i) = j$. Let $v_{i+1}$ be the successor of $v_i$ such that $(v_i, v_{i+1})$ is an edge of $G_m$ and $FullLabel(v_{i+1})$ is satisfied at the state $u_{f(i+1)}$ of $M$. Such a node $v_{i+1}$ exists because $FullLabel(v_i)$ is satisfied at the state $u_{f(i)} = u_j = u_{f(i+1)}$ of $M$. By setting $i := i + 1$, all the invariants (4)-(6) still hold.
– Case \( \rho \) has the principal formula of the form \( \langle A', q' \rangle \varphi' \): During a sequence of applications of static rules between two applications of the transitional rule, proceed as for realizing \( \langle A', q' \rangle \varphi' \) in \( G_m \) (like for the current \( \bowtie \)-realization of \( \langle A, q \rangle \varphi \) in \( G_m \) at \( v_0 \)). This decides how to choose \( v_{i+1} \) and has effects on terminating the trace (to obtain a \( \bowtie \)-realization for \( \langle A, q \rangle \varphi \) in \( G_m \) at \( v_0 \)). We also choose \( \varphi_{i+1} = \varphi_i \) and \( f(i+1) = f(i) = j \). By setting \( i := i + 1 \), all the invariants [4]-[6] still hold.

4. Case \( v_i \) is expanded using the transitional rule:

Since \( v_i \) is a state and \( \varphi_i \neq \varphi \), by the invariants [5] and [6], we must have that \( \varphi_i = \langle \omega j+1 \rangle \varphi \) with \( \omega j+1 \in \Sigma \). Let \( v_{i+1} \) be the successor of \( v_i \) with \( CELabel(v_{i+1}) = \varphi_i \).

Let \( \varphi_{i+1} = \langle A, q_j+1 \rangle \varphi \) and \( f(i+1) = j + 1 \). Clearly, the invariant [3] holds for \( i+1 \). By [2], \( \varphi_{i+1} \) is satisfied at the state \( u_{j+1} \) of \( M \). By the invariant [4], the other formulas of \( Label(v_{i+1}) \) are also satisfied at the state \( u_{j+1} \) of \( M \). That is, the invariant [4] holds for \( i+1 \). Clearly, the invariants [5] and [6] remain true after increasing \( i \) by 1. So, by setting \( i := i + 1 \), all the invariants [3]-[6] still hold.

We now show that the loop terminates. Observe that any sequence of applications of static rules that contribute to the trace \( (v_0, \varphi_0), \ldots, (v_i, \varphi_i) \) of \( \langle A, q \rangle \varphi \) in \( G_m \) eventually ends because:

– each formula not of the forms \( \langle A', q' \rangle \varphi' \) or \( \langle \psi'? \rangle \varphi' \) may be reduced at most once
– each formula of the form \( \langle \psi''? \rangle \varphi' \) is reduced to \( \psi' \) and \( \varphi' \)
– each formula of the form \( \langle A', q' \rangle \varphi' \) is reduced according to some \( \bowtie \)-realization.

That is, sooner or later either \( \varphi_i = \varphi \) or \( v_i \) is a node that is expanded by the transitional rule. In the second case, if \( f(i) = j \) then \( f(i+1) = j + 1 \). As the image of \( f \) is \( \{0, \ldots, k\} \), the construction of the trace must end at some step (with \( \varphi_i = \varphi \)) and we obtain a \( \bowtie \)-realization in \( G_m \) for \( \langle A, q \rangle \varphi \) at \( v_0 \).

4.2 Model Graphs

We will prove completeness of \( CL \) via model graphs [25][10][19][22][21].

**Definition 4.5.** A model graph (also known as a Hintikka structure) is a tuple \( \langle W, R, H \rangle \), where \( W \) is a set of nodes, \( R \) is a mapping that associates each \( \sigma \in \Sigma \) with a binary relation \( R_{\sigma} \) on \( W \), and \( H \) is a mapping that associates each node of \( W \) with a set of formulas.

We use model graphs merely as data structures, but we are interested in consistent and saturated model graphs as defined below. Model graphs differ from “and-or” graphs in that a model graph contains only “and”-nodes and its edges are labeled by simple programs. Roughly speaking, given an “and-or” graph \( G \) with a consistent matching \( G_m \), to construct a model graph one can stick together the nodes in a “saturation path” of a node of \( G_m \) to create a node for the model graph. Details will be given later.

A trace of a formula \( \langle A, q \rangle \varphi \) at a node in a model graph is defined analogously as for the case of “and-or” graphs, as stated in the following definition.

**Definition 4.6.** Given a model graph \( M = \langle W, R, H \rangle \) and a node \( v \in W \), a trace of a formula \( \langle A, q \rangle \varphi \in H(v) \) (starting from \( v \)) is a sequence \( (v_0, \varphi_0), \ldots, (v_k, \varphi_k) \) such that:

– \( v_0 = v \) and \( \varphi_0 = \langle A, q \rangle \varphi \),
– for every \( 1 \leq i \leq k \), \( \varphi_i \in H(v_i) \),
– for every \( 1 \leq i \leq k \), if \( v_i = v_{i-1} \) then:
  • \( \varphi_{i-1} \) is of the form \( \langle A, q' \rangle \varphi \) and \( \varphi_i = \langle \omega \rangle \langle A, q'' \rangle \varphi \) for some \( \omega \) and \( q'' \) such that \( \langle q', \omega, q'' \rangle \in \delta_A \), or
  • \( \varphi_{i-1} \) is of the form \( \langle A, q' \rangle \varphi \) with \( q' \in F_A \), \( \varphi_i = \varphi \), and \( i = k \), or
  • \( \varphi_{i-1} \) is of the form \( \langle \psi'? \rangle \langle A, q' \rangle \varphi \) and \( \varphi_i = \langle A, q' \rangle \varphi \)
for every $1 \leq i \leq k$, if $v_i \neq v_{i-1}$ then $\varphi_{i-1}$ is of the form $\langle \sigma \rangle (A, q') \varphi$, $\varphi \in \Sigma, \varphi_{i} = (A, q') \varphi$, and $(v_{i-1}, v_i) \in R_\sigma$.

A trace $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ of $(A, q) \varphi$ in a model graph $M$ is called a $\Diamond$-realization for $(A, q) \varphi$ at $v_0$ if $\varphi_k = \varphi$.

Similarly as for markings of “and-or” graphs, we define the consistency of a model graph as follows.

**Definition 4.7.** A model graph $M = \langle W, R, H \rangle$ is consistent if:

- local consistency: for every $v \in W$, $H(v)$ contains neither $\bot$ nor a clashing pair of the form $p$ and $\neg p$; and
- global consistency: for every $v \in W$, every formula $(A, q) \varphi$ of $H(v)$ has a $\Diamond$-realization (at $v$).

**Definition 4.8.** A model graph $M = \langle W, R, H \rangle$ is said to be CL-saturated if the following conditions hold for every $v \in W$ and every $\varphi \in H(v)$:

- if $\varphi = \psi \land \xi$ then $\{\psi, \xi\} \subset H(v)$
- if $\varphi = \psi \lor \xi$ then $\psi \in H(v)$ or $\xi \in H(v)$
- if $\varphi = [\alpha] \psi$ and $I_{\alpha v} = \{q_1, \ldots, q_k\}$ then $\{[A_{\alpha}, q_1] \psi, \ldots, [A_{\alpha}, q_k] \psi\} \subset H(v)$
- if $\varphi = [A, q] \psi$ and $\delta(A, q) = \{\omega_1, q_1, \ldots, (\omega_k, q_k)\}$ then $\{\square_{\omega_1}[A, q_1] \psi, \ldots, \square_{\omega_k}[A, q_k] \psi\} \subset H(v)$
- if $\varphi = [A, q] \psi$ and $q \in F_A$ then $\psi \in H(v)$
- if $\varphi = \square(\xi) \psi$ then $\xi \in H(v)$ or $\psi \in H(v)$
- if $\varphi = (\alpha) \psi$, $\alpha \notin \Sigma$, $\alpha$ is not a test and $I_{\alpha v} = \{q_1, \ldots, q_k\}$ then one of the formulas $(\alpha_{\alpha}, q_1) \psi, \ldots, (\alpha_{\alpha}, q_k) \psi$ belongs to $H(v)$
- if $\varphi = (\xi?) \psi$ then $\xi \in H(v)$
- if $\varphi = (\sigma) \psi$ then there exists $w$ such that $R_{\sigma}(v, w)$ and $\psi \in H(w)$
- if $\varphi = \square_{\sigma} \psi$ and $R_{\sigma}(v, w)$ holds then $\psi \in H(w)$
- if $\varphi = \square_{\sigma} \psi$ and $R_{\sigma}(u, v)$ holds then $\psi \in H(u)$.

**Definition 4.9.** Given a model graph $M = \langle W, R, H \rangle$, the $L$-model corresponding to $M$ is the Kripke model $M'$ such that $\Delta^{M'} = W$, $\mu^{M'} = \{w \in W \mid p \in H(w)\}$ for $p \in \text{PROP}$, $g^{M'} = R_{\varphi}$ for $\sigma \in \Sigma^+$, where $R_{\sigma}$ for $\sigma \in \Sigma$ are the smallest binary relations on $W$ such that:

- $R_{\sigma} \subseteq R_{\sigma}'$ and $R_{\sigma}' = (R_{\sigma})^{-1}$ for every $\sigma \in \Sigma$, and
- if $\sigma \rightarrow q_1, \ldots, q_k \in S$, where $S$ is the symmetric regular semi-Thue system of $L$, then $R_{q_1} \circ \cdots \circ R_{q_k} \subseteq R_{\sigma}'$.

The smallest binary relations mentioned in the above definition exist because:

- $R_{\sigma} = (R_{\sigma})^{-1}$ iff $(R_{\sigma})^{-1} \subseteq R_{\sigma}$ and $(R_{\sigma})^{-1} \subseteq R_{\sigma}'$.
- an expression of the form $R^{-1} \subseteq R'$ is equivalent to $\forall x, y (R(y, x) \rightarrow R'(x, y))$
- an expression of the form $R_1 \circ \cdots \circ R_k \subseteq R'$ is equivalent to $\forall x_0, \ldots, x_h (R_1(x_0, x_1) \land \cdots \land R_h(x_{h-1}, x_h) \rightarrow R'(x_0, x_h))$
- a set of first-order program clauses like the ones listed above is called a Datalog program and always has the smallest model.

Define the NCNF of $\neg \square_{\sigma} \varphi$ to be $(\sigma)[\varphi]$. Recall that the NCNF of $\neg [\alpha] \varphi$, $\neg (\alpha) \varphi$, $\neg [A, q] \varphi$, $\neg (A, q) \varphi$ are $(\alpha)[\varphi]$, $[\alpha][\varphi]$, $(A, q)[\varphi]$, $[A, q][\varphi]$, respectively.

---

7 The condition $\psi \in H(v)$ is taken care of by the global consistency.
8 Note that the symmetry of $S$ is essential here.
Lemma 4.10. Let $M = (W, R, H)$ be a consistent and CL-saturated model graph and let $M'$ be the L-model corresponding to $M$. Then, for any $w \in W$, if $M', w \models \varphi$ or $\varphi \in H(w)$ then $\varphi \notin H(w)$.

Proof. By induction on the structure of $\varphi$, using the global consistency. <

Lemma 4.11. Let $\alpha$ be a program, $q \in Q_{A,\alpha}$, and $\gamma \in L((A, q))$. If the rule $\sigma \rightarrow q_1 \ldots q_k$ belongs to $S$ then replacing any occurrence of $\sigma$ in $\gamma$ by $q_1 \ldots q_k$ results in $\gamma' \in L((A, q))$.

Proof. This lemma follows from the observation that: if $\sigma \rightarrow q_1 \ldots q_k \in S$, $q \in \Sigma$, and $\gamma_2 \in L(A_q)$, then replacing any occurrence of $\sigma$ in $\gamma_2$ by $q_1 \ldots q_k$ results in $\gamma'_2 \in L(A_q)$. <

The following lemma is the main result of this subsection.

Lemma 4.12. Let $X$ and $\Gamma$ be finite sets of formulas in NCNF of the base language, and let $M = (W, R, H)$ be a consistent and CL-saturated model graph such that $\Gamma \subseteq H(w)$, for all $w \in W$, and $X \subseteq H(\tau)$, for some $\tau \in W$. Then the L-model $M'$ corresponding to $M$ validates $\Gamma$ and satisfies $X$ at $\tau$.

Proof. Observer that:

1. if $[A, q] \psi \in H(w)$ and $R_A(w, w')$ and $\delta_A(q, \sigma, q')$ then $[A, q'] \psi \in H(w')$ (7)
2. if $[A, q] \psi \in H(w)$ and $R_A(-w, w')$ and $\delta_A(q, \sigma, q')$ then $[A, q'] \psi \in H(w')$. (8)

These assertions hold because:

1. if $[A, q] \psi \in H(w)$ and $\delta_A(q, \sigma, q')$ holds then $\square_\sigma [A, q'] \psi \in H(w)$
2. if $\square_\sigma [A, q'] \psi \in H(w)$ and $R_A(w, w')$ holds then $[A, q'] \psi \in H(w')$
3. if $\square_\sigma [A, q'] \psi \in H(w)$ and $R_A(-w, w')$ holds then $[A, q'] \psi \in H(w')$.

Using induction on the construction of $\varphi$, we now prove that for any $u \in W$, if $\varphi \in H(u)$ and $\varphi$ is a formula of the base language then $M', u \models \varphi$. Assume that $\varphi \in H(u)$. The only non-trivial cases are when

1. $\varphi = [\alpha] \psi$, or $\varphi = \langle \alpha \rangle \psi$, where $\alpha \notin \Sigma$ and $\alpha$ is not a test. (9)
2. Consider the case (9) and let $q \in I_{A, \alpha}$. By Lemma 3.1, it suffices to show that $M', u \models [A, q] \psi$. Since $M$ is CL-saturated and $\varphi \in H(u)$, we have that $[A, q] \psi \in H(u)$. Let $v \in W$ be a node such that $(u, v) \in (A, q)^{M'}$. We show that $\psi \in H(v)$. By the construction of $M'$ and Lemma 4.11, there exist a word $\omega_1 \ldots \omega_k \in L((A_q, q))$ and elements $w_0, \ldots, w_k$ of $W$ such that $w_0 = u$, $w_k = v$, and for every $1 \leq i \leq k$:

3. $\omega_i \in \Sigma$ and $R_{w_i}(w_{i-1}, w_i)$ or $R_{w_i}'(w_i, w_{i-1})$ holds, or $\omega_i$ is of the form $(\xi_i, ?)$ and $w_{i-1} = w_i \in \xi_i'$. (10)

Let $q_0 = q, q_1, \ldots, q_k$ be an accepting run of $(A, q)$ on the word $\omega_1 \ldots \omega_k$. We have that $q_k \in F_{A, q}$. We prove by an inner induction on $0 \leq i \leq k$ that $[A, q_i] \psi \in H(w_i)$. The base case $i = 0$ clearly holds. Inductively, assume that $1 \leq i \leq k$ and $[A, q_{i-1}] \psi \in H(w_{i-1})$. If (11) holds then, by (7) and (8), it follows that $[A, q_i] \psi \in H(w_i)$. Suppose that (12) holds. Since $[A, q_{i-1}] \psi \in H(w_{i-1})$ and $M$ is $CL$-saturated, we have that $\square_\sigma [A, q_i] \psi \in H(w_{i-1})$.

Since $M', w_{i-1} \models \xi_i$, by Lemma 4.10, $\xi_i \notin H(w_{i-1})$. Since $M$ is $CL$-saturated, it follows that $[A, q_i] \psi \in H(w_i)$.

Therefore $M', w_{i-1} \models \varphi$. Since $w_{i-1} = w_i$, we also have that $[A, q_i] \psi \in H(w_i)$. This completes the inner induction. As a consequence, $[A, q_k] \psi \in H(w_k)$. Since $q_k \in F_{A, q}$ and $v = w_k$, it follows that $\psi \in H(v)$. By the inductive assumption, we have that $M', v \models \psi$. Therefore $M', u \models [A, q] \psi$. 

Consider the case (10). Since $M$ is $CL$-saturated, there exists $q \in I_{h_0}$ such that $\langle h_0, q, \psi \rangle \in H(u)$. By Lemma 3.1 it suffices to show that $M', u |\models \langle h_0, q, \psi \rangle$. Since $\langle h_0, q, \psi \rangle \in H(u)$, by the global consistency of $M$, there exist an accepting run $q_0 = q, q_1, \ldots, q_k$ of $\langle h_0, q, \psi \rangle$ on a word $\omega_1 \ldots \omega_k$ and nodes $w_0 = u, w_1, \ldots, w_k$ of $M$ such that $\psi \in H(w_k)$ and, for $1 \leq i \leq k$, $\langle h_0, q_i, \psi \rangle \in H(w_i)$ and if $\omega_i \in \Sigma$ then $R_{\omega_i}(w_i, w_{i-1})$ holds, else $\omega_i$ is of the form $\langle \xi_i, ? \rangle$ and $w_{i-1} = w_i$ and $\langle \xi_i, ? \rangle \in H(w_i)$. Since $M$ is $CL$-saturated, in the case $\omega_i = \langle \xi_i, ? \rangle$ we have that $\xi_i \in H(w_{i-1})$, and by the inductive assumption, it follows that $M', w_{i-1} \models \xi_i$. Hence $(w_0, w_k) \in \langle h_0, q, \psi \rangle$. Since $\psi \in H(w_k)$, by the inductive assumption, we have that $M', w_k \models \psi$. Hence $M', w_0 \models \langle h_0, q, \psi \rangle$. That is, $M', u \models \langle h_0, q, \psi \rangle$. This completes the proof. \hfill \(\diamondsuit\)

4.3 Completeness

**Definition 4.13.** Let $G$ be a $CL$-tableau for $(X, \Gamma)$, $G_m$ be a consistent marking of a $G$, and $v$ be a node of $G_m$. A saturation path of $v$ w.r.t. $G_m$ is a finite sequence $v_0 = v, v_1, \ldots, v_k$ of nodes of $G_m$, with $k \geq 0$, such that, for every $0 \leq i < k$, $v_i$ is a non-state and $(v_i, v_{i+1})$ is an edge of $G_m$, and $v_k$ is a state.

**Lemma 4.14.** Let $G$ be a $CL$-tableau for $(X, \Gamma)$ with a consistent marking $G_m$. Then each node $v$ of $G_m$ has a saturation path w.r.t. $G_m$.

**Proof.** We construct a saturation path $v_0, v_1, \ldots$ of $v$ w.r.t. $G_m$ as follows. Set $v_0 := v$ and $i := 0$. While $v_i$ is not a state do:

- If the principal of the static rule expanding $v_i$ is not of the form $\langle A, q, \varphi \rangle$ then let $v_{i+1}$ be any successor of $v_i$ that belongs to $G_m$ and set $i := i + 1$.
- If the principal of the static rule expanding $v_i$ is of the form $\langle A, q, \varphi \rangle$ then:
  - let $v_{i+1}, \ldots, v_j$ be the longest sequence of non-states of $G_m$ such that there exist formulas $\varphi_{i+1}, \ldots, \varphi_j$ such that the sequence $(v_i, \varphi_i), \ldots, (v_j, \varphi_j)$ is a prefix of a (fixed) $\Diamond$-realization in $G_m$ for $\langle A, q, \varphi \rangle$ at $v_i$;
  - set $i := j$.

The loop terminates because:

- each formula not of the forms $\langle A, q, \varphi \rangle$ and $\langle \psi, ? \rangle$ may be reduced at most once
- each formula of the form $\langle \psi, ? \rangle$ is reduced to $\psi$ and $\varphi$
- each formula of the form $\langle A, q, \varphi \rangle$ is reduced according to some $\Diamond$-realization. \hfill \(\diamondsuit\)

We are now in position to prove completeness of the calculus.

**Lemma 4.15 (Completeness).** Let $G$ be a $CL$-tableau for $(X, \Gamma)$. Suppose that $G$ has a consistent marking $G_m$. Then $X$ is $L$-satisfiable w.r.t. the set $\Gamma$ of global assumptions.

**Proof.** We construct a model graph $M = \langle W, R, H \rangle$ as follows:

Let $v_0$ be the root of $G_m$ and $v_0, \ldots, v_k$ be a saturation path of $v_0$ w.r.t. $G_m$. Set $R_{\sigma} := \emptyset$ for all $\sigma \in \Sigma$ and set $W := \{\tau\}$, where $\tau$ is a new node. Set further $H(\tau) := Label(v_k) \cup RFmls(v_k)$.

Mark $\tau$ as unresolved and let $f(\tau) = v_k$. (Each node of $M$ will be marked either as unresolved or as resolved, and $f$ will map each node of $M$ to a state of $G_m$.)

While $W$ contains unresolved nodes, take one unresolved node $w_0$ and do:

1. For every formula $\langle \sigma, q \rangle \in H(w_0)$ (with $\sigma \in \Sigma$) do:
   - (a) Let $\varphi_0 = \langle \sigma, q \rangle$ and $\varphi_1 = \varphi$.
   - (b) Let $u_0 = f(w_0)$ and let $u_1$ be the successor of $u_0$ such that $CELlabel(u_1) = \varphi_0$. (As a maintained property of $f$, $u_0$ is a state, $\varphi_0 \in Label(u_0)$, and therefore $AfterTrans(u_1)$ holds and $\varphi_1 \in Label(u_1)$.)
(c) If \( \varphi \) is of the form \( \langle A, q \rangle \psi \) then let \( (u_1, \varphi_1), \ldots, (u_t, \varphi_t) \) to be a \( \diamond \)-realization in \( G_m \) for \( \varphi_1 \) at \( u_1 \) and let \( u_1, \ldots, u_m \) to be a saturation path of \( u_t \) w.r.t. \( G_m \).

(d) Else let \( u_1, \ldots, u_m \) be a saturation path of \( u_t \) w.r.t. \( G_m \).

(e) Let \( j_0 = 0 < j_1 < \ldots < j_{n-1} < j_n = m \) be all the indices such that, for \( 0 \leq j \leq m \), \( u_j \) is a state of \( G \) iff \( j \in \{ j_0, \ldots, j_n \} \). For \( 0 \leq s \leq n - 1 \), let \( \langle \sigma_s \rangle \varphi_{j_s+1} = CELabel(u_{j_s+1}) \). (We have that \( \sigma_0 = \sigma \).)

(f) For \( 1 \leq s \leq n \) do:
   
   i. Let \( Z_s = \{ \langle \text{Label}(u_i) \cup RFmls(u_i) \rangle \mid j_{s-1} + 1 \leq i \leq j_s \} \).
   
   ii. If there does not exist \( w_s \in W \) such that \( H(w_s) = Z_s \) then: add a new node \( w_s \) to \( W \), set \( H(w_s) := Z_s \), mark \( w_s \) as unresolved, and let \( f(w_s) = u_{j_s} \). 
   
   iii. Add the pair \( (w_{s-1}, w_s) \) to \( R_{\sigma_{s-1}} \).

2. Mark \( w_0 \) as resolved.

As \( H \) is a one-to-one function and \( H(w) \) of each \( w \in W \) is a subset of \( \text{closure}_L(X \cup \Gamma) \), the above construction terminates and results in a finite model graph.

Observe that, in the above construction we transform the chain \( u_0, \ldots, u_m \) of nodes of \( G_m \) to a chain \( u_0, \ldots, u_n \) of nodes of \( M \) by sticking together nodes in every saturation path. Hence, \( M \) is \( \mathcal{CL} \)-saturated and satisfies the local consistency property.

Suppose that \( \langle A', q' \rangle \psi' \in H(w_0) \). Since \( u_0 \) is a state of the consistent marking \( G_m \), the formula \( \langle A', q' \rangle \psi' \) of \( H(w_0) \) has a (finite) trace starting from \( w_0 \), using only \( w_0 \) (as the first component of the pairs), and ending with a pair of the form \( (u_0, \psi') \) or \( (w_0, \langle \sigma' \rangle \langle A', q'' \rangle \psi') \). This together with Step 1 implies that \( M \) satisfies the global consistency property. Hence, \( M \) is a consistent and \( \mathcal{CL} \)-saturated model graph.

Since \( \text{Label}(v_0) = X \cup \Gamma \) and \( \text{Label}(v_0) \subseteq \text{Label}(v_k) \cup RFmls(v_k) \), we have that \( X \subseteq H(\tau) \) and \( \Gamma \subseteq H(\tau) \). Consider Step 1 of the construction, as \( u_{j_{s-1}} \) is an “and”-node and \( u_{j_{s-1}+1} \) is a successor of \( u_{j_{s-1}} \) that is created by the transitional rule, we have that \( \text{Label}(u_{j_{s-1}+1}) \supseteq \Gamma \), and hence \( \text{Label}(u_{j_{s-1}}) \cup RFmls(u_{j_{s-1}}) \supseteq \text{Label}(u_{j_{s-1}+1}) \supseteq \Gamma \). Hence \( \Gamma \subseteq H(w_s) \) for every \( w_s \in W \).

By Lemma 4.12, the Kripke model corresponding to \( M \) validates \( \Gamma \) and satisfies \( X \) at \( \tau \). Hence, \( X \) is \( L \)-satisfiable w.r.t. \( \Gamma \).

5 An ExpTime Tableau Decision Procedure for CPDL_{reg}

Let \( G = (V, E, \nu) \) be a \( \mathcal{CL} \)-tableau for \( (X, \Gamma) \) and \( G_m \) be a marking of \( G \). The graph \( G_t \) of traces of \( G_m \) in \( G \) is defined as follows:

- nodes of \( G_t \) are pairs \( (v, \varphi) \), where \( v \in V \) and \( \varphi \in \text{Label}(v) \)
- a pair \( ((v, \varphi), (w, \psi)) \) is an edge of \( G_t \) if \( v \) is a node of \( G_m \), \( \varphi \) is of the form \( \langle A, q \rangle \xi \) or \( \langle \omega \rangle \langle A, q \rangle \xi \), and the sequence \( (v, \varphi), (w, \psi) \) is a subsequence of a trace in \( G_m \).

A node \( (v, \varphi) \) of \( G_t \) is an end node if \( \varphi \) is a formula of the base language. A node of \( G_t \) is productive if there is a path connecting it to an end node.

Algorithm 1 (given on page 22) is a simple EXPTime algorithm for checking \( L \)-satisfiability of \( X \) w.r.t. \( \Gamma \).

Lemma 5.1. Let \( h = |\Sigma| \) and let \( k \) be the sum of the sizes of the automata \( A_\sigma \) (for \( \sigma \in \Sigma \)), and \( l \) be the size of \( X \cup \Gamma \). Then Algorithm 1 for \( X, \Gamma \), \( A \) runs in \( 2^{O(n)} \) steps, where \( n \) be the size of \( \text{closure}_L(X \cup \Gamma) \) and is polynomial in \( k, h, l \).

Proof. By Lemma 3.3, \( n \) is polynomial in \( k, h, l \) and the graph \( G \) has \( 2^{O(n)} \) nodes and can be constructed in \( 2^{O(n)} \). For each node \( v \) of \( G \), \( \text{Label}(v) \subseteq \text{closure}_L(X \cup \Gamma) \). Constructing graphs of traces and updating statuses of nodes can be done in polynomial time in the size of \( G \). Hence, totally, Algorithm 1 runs in \( 2^{O(n)} \) steps.

<
Algorithm 1: for checking \( L \)-satisfiability of \( X \) w.r.t. \( \Gamma \).

**Input:** finite sets \( X \) and \( \Gamma \) of formulas in NCNF of the base language, the mapping \( A \) specifying the finite automata of the symmetric regular semi-Thue system of the considered CPDL\(_{reg} \) logic \( L \)

**Output:** true if \( X \) is \( L \)-satisfiable w.r.t. \( \Gamma \), and false otherwise

1. Let \( G = (V, E, \nu) \) be the CL-tableau constructed by Tableau\((X, \Gamma)\) (using \( A \));
2. while \( \text{Status}(v) \neq \text{unsat} \) do
   3. let \( G_m \) be the subgraph of \( G \) induced by the nodes with status different from \text{unsat} and \text{incomplete};
   4. /* we have that \( G_m \) is a marking of \( G \) */
   5. construct the graph \( G_t \) of traces of \( G_m \) in \( G \);
   6. if there exist a node \( v \) of \( G_m \) and a formula \( \langle A, q, \varphi \rangle \in \text{Label}(v) \) such that \( \langle v, \langle A, q, \varphi \rangle \rangle \) is not a productive node of \( G_t \), then
   7. \( \text{PropagateStatus}(v) \)
   8. else return \text{true};

return \text{false}.

**Theorem 5.2.** Let \( S \) be a symmetric regular semi-Thue system over \( \Sigma \), \( A \) be the mapping specifying the finite automata of \( S \), and \( L \) be the CPDL\(_{reg} \) logic corresponding to \( S \). Let \( X \) and \( \Gamma \) be finite sets of formulas in NCNF of the base language. Then Algorithm 2 is an ExpTime (optimal) decision procedure for checking whether \( X \) is \( L \)-satisfiable w.r.t. the set \( \Gamma \) of global assumptions.

**Proof.** It is straightforward to prove by induction that the algorithm has the invariant that a consistent marking of \( G \) cannot contain any node with status \text{unsat}. For the base step, use Theorem 3.3 and the local consistency property. For the induction step, use Theorem 5.4 and the global consistency property.

The algorithm returns \text{false} only when \( \text{Status}(v) = \text{unsat} \), i.e., only when \( G \) does not have any consistent marking. At Step 7, \( G_m \) is a consistent marking of \( G \). That is, the algorithm returns \text{true} only when \( G \) has a consistent marking. Therefore, by Theorem 3.4, Algorithm 1 is a decision procedure for the considered problem. The complexity was established by Lemma 5.1.

Clearly, one may modify procedure Tableau to make it stop as soon as \( \text{Status}(v) = \text{unsat} \). We excluded this condition just to formalize and prove Theorem 3.4 but it does not affect correctness of Theorem 5.2.

## 6 Checking Global Consistency On-the-Fly

Observe that Algorithm 1 first constructs a CL-tableau and then checks whether the tableau contains a consistent marking. To speed up the performance these two tasks can be done concurrently. For this we can update \( G_m \) and \( G_t \) during the construction of \( G \) and detect and propagate \text{unsat} (and \text{sat}) on-the-fly using both the local and global consistency properties as discussed below.

For nodes \( \langle v, \varphi \rangle \) of the graph of traces \( G_t \), define \( \text{SemiEndNodes}(v, \varphi) \) to be the smallest sets of nodes of \( G_t \) that satisfy the following conditions:

- if \( \langle v, \varphi \rangle \) is an end node of \( G_t \) or \( \text{Status}(v) \in \{\text{unexpanded, sat}\} \) then
  \[ \text{SemiEndNodes}(v, \varphi) = \{\langle v, \varphi \rangle\} \],
- else \( \text{SemiEndNodes}(v, \varphi) = \bigcup \{\text{SemiEndNodes}(w, \psi) \mid (w, \psi) \text{ is a successor of } \langle v, \varphi \rangle \text{ in } G_t \text{ different from } \langle v, \varphi \rangle\} \setminus \{(v, \varphi)\} \).

Observe that, during the construction of \( G \) and \( G_t \), if \( \text{SemiEndNodes}(v, \varphi) = \emptyset \) then \( \langle v, \varphi \rangle \) is not and will never be a productive node of \( G_t \) and hence \( \text{Status}(v) \) can be set to \text{unsat}. Furthermore, at the end (i.e., when \( G \) is a full “and-or” graph, \( G_m \) is the subgraph of \( G \) induced...
by the nodes with status different from \textit{unsat} and \textit{incomplete}, and \( G_t \) is the graph of traces of \( G_m \) in \( G \), if a node \((v, \varphi)\) of \( G_t \) has \( \text{SemiEndNodes}(v, \varphi) \neq \emptyset \) then \((v, \varphi)\) is a productive node of \( G_t \).

An improved \textsc{ExpTime} tableau decision procedure for checking \( L \)-satisfiability of \( X \) w.r.t. \( \Gamma \) as follows. During the construction of a \textsc{Cl}-tableau for \((X, \Gamma)\):

- let \( G = (V, E, \nu) \) be the current (partial) “and-or” graph;
- update \( G_m \) on-the-fly so that it is the subgraph of \( G \) induced by the nodes with status different from \textit{unsat} and \textit{incomplete};
- update \( G_t \) on-the-fly so that it is the graph of traces of \( G_m \) in \( G \);
- update \( \text{SemiEndNodes}(v, \varphi) \) on-the-fly for nodes \((v, \varphi)\) of \( G_t \) as follows:
  - when a new node \( v \) with status \textit{unexpanded} is created for \( G \), for every node \((v, \varphi)\) created for \( G_t \), set \( \text{SemiEndNodes}(v, \varphi) := \{(v, \varphi)\} \);
  - when a node \( v \) of \( G \) receives status \textit{sat}, for every node \((v, \varphi)\) of \( G_t \), set \( \text{SemiEndNodes}(v, \varphi) := \{(v, \varphi)\} \);
  - when a node \( v \) of \( G \) is expanded and receives status \textit{expanded},
    - for every end node \((v, \varphi)\) of \( G_t \), set \( \text{SemiEndNodes}(v, \varphi) := \{(v, \varphi)\} \);
    - for every non-end node \((v, \varphi)\) of \( G_t \), set
      \[
      \text{SemiEndNodes}(v, \varphi) := \bigcup \{ \text{SemiEndNodes}(w, \psi) \mid (w, \psi) \text{ is a successor of } (v, \varphi) \text{ in } G_t \text{ different from } (v, \varphi) \} \setminus \{(v, \varphi)\};
      \]
  - when a node \((w, \psi)\) of \( G_t \) has \( \text{SemiEndNodes}(w, \psi) \) changed, for every predecessor \((v, \varphi)\) of \((w, \psi)\) in \( G_t \), update \( \text{SemiEndNodes}(v, \varphi) \) by (13);
  - when a node \( w \) of \( G \) receives status \textit{unsat} or \textit{incomplete} or is deleted, for every node \((v, \varphi)\) of \( G_t \) that is a predecessor of some \((w, \psi)\), update \( \text{SemiEndNodes}(v, \varphi) \) by (13);
    - when a node \((v, \varphi)\) of \( G_t \) receives \( \text{SemiEndNodes}(v, \varphi) = \emptyset \), set \( \text{Status}(v) := \text{unsat} \) and execute \text{PropagateStatus}(v);
    - when the root \( \nu \) of \( G \) receives status \textit{unsat}, return false (which means \( X \) is not \( L \)-satisfiable w.r.t. \( \Gamma \));
    - at the end (not terminated as in the above case), return true (which means \( X \) is \( L \)-satisfiable w.r.t. \( \Gamma \)).

See also [20] for other possible optimization techniques, which have been implemented and evaluated for the description logic \( \mathcal{ALC} \).

7 Conclusions

As discussed in the introduction, translating the general satisfiability checking problem in \( \text{CPDL}_{\mathcal{Reg}} \) into the satisfiability checking problem in \( \text{CPDL} \) may increase the complexity (and may decrease efficiency). Therefore, the direct approach for automated reasoning in \( \text{CPDL}_{\mathcal{Reg}} \) is worth studying.

In this paper we have given a tableau calculus and the first cut-free \textsc{ExpTime} tableau decision procedure for checking satisfiability in the logic \( \text{CPDL}_{\mathcal{Reg}} \). Our decision procedure uses global caching for states and local caching for non-states in tableaux. In comparison with the tableau calculus given in our joint papers [8,9] for \( \text{CPDL}_{\mathcal{Reg}} \), the tableau calculus given in the current paper is essentially better since it is cut-free. Furthermore, we have explicitly incorporated on-the-fly propagation of local (in)consistency into the latter calculus and the corresponding decision procedure. We have also discussed how on-the-fly propagation of global (in)consistency can be incorporated into the tableau decision procedure.

When restricted to \( \text{REG}^c \), our tableau decision procedure is much better than the one proposed by us and Szalas in [22] for \( \text{REG}^c \), as it does not use cuts and it does not have to check the global consistency property (this is an effect of cut elimination).
On technical matters, apart from global caching of states as in the works \cite{14,15} by Goré and Widmann, we allow to cache also non-states. We apply global caching for nodes in the local graphs of non-states $v$ satisfying $\text{AfterTrans}(v)$. Such local graphs in \cite{14,15} are trees. We propose not to delay solving incompatibility w.r.t. converse as long as in \cite{14,15}. One can solve incompatibility w.r.t. converse as soon as possible as done in this paper. By giving a higher priority to the current branch even when it involves converse we can further favor depth-first search.

References

1. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. Studia Logica, 69:5–40, 2001.
2. G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. Information and Computation, 162(1-2):117–137, 2000.
3. S. Demri. The complexity of regularity in grammar logics and related modal logics. Journal of Logic and Computation, 11(6):933–960, 2001.
4. S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. arXiv:cs.LO/0306117, 2004.
5. S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. Journal of Logic, Language and Inform., 14(3):289–329, 2005.
6. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. A framework for graded beliefs, goals and intentions. Fundam. Inform., 100(1-4):53–76, 2010.
7. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. Tractable approximate knowledge fusion using the Horn fragment of serial propositional dynamic logic. Int. J. Approx. Reasoning, 51(3):346–362, 2010.
8. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. Converse-PDL with regular inclusion axioms: a framework for MAS logics. Journal of Applied Non-Classical Logics, 21(1):61–91, 2011.
9. G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In L.C. Aiello, J. Doyle, and S.C. Shapiro, editors, Proceedings of KR’1996, pages 316–327, Morgan Kaufmann, 1996.
10. R. Goré. Tableau methods for modal and temporal logics. In D’Agostino et al, editor, Handbook of Tableau Methods, pages 297–396. Kluwer, 1999.
11. R. Goré and L.A. Nguyen. A tableau system with automaton-labelled formulae for regular grammar logics. In B. Beckert, editor, Proceedings of TABLEAUX 2005, LNAI 3702, pages 138–152. Springer-Verlag, 2005.
12. R. Goré and L.A. Nguyen. Analytic cut-free tableaux for regular modal logics of agent beliefs. In F. Sadri and K. Satoh, editors, Proceedings of CLIMA VIII, pages 274–289, 2007.
13. R. Goré and L.A. Nguyen. EXPTIME tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In N. Olivetti, editor, Proc. of TABLEAUX 2007, LNAI 4548, pages 133–148. Springer-Verlag, 2007.
14. R. Goré and F. Widmann. Sound global state caching for $\text{ALC}$ with inverse roles. In M. Giese and A. Waaler, editors, Proceedings of TABLEAUX 2009, volume 5607 of LNCS, pages 205–219. Springer, 2009.
15. R. Goré and F. Widmann. Optimal and cut-free tableaux for propositional dynamic logic with converse. In J. Giesl and R. Hähnle, editors, Proceedings of IJCAR 2010, volume 6173 of LNCS, pages 225–239. Springer, 2010.
16. D. Harel, D. Kozen, and J. Tiuryn. Dynamic Logic. MIT Press, 2000.
17. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\text{SROIQ}$. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, Proc. of KR’2006, pages 57–67. AAAI Press, 2006.
18. A. Mateescu and A. Salomaa. Formal languages: an introduction and a synopsis. In Handbook of Formal Languages - Volume 1, pages 1–40. Springer, 1997.
19. L.A. Nguyen. Analytic tableau systems and interpolation for the modal logics KB, KDB, K5, KD5. Studia Logica, 69(1):41–57, 2001.
20. L.A. Nguyen. An efficient tableau prover using global caching for the description logic $\text{ALC}$. Fundamenta Informaticae, 93(1-3):273–288, 2009.
21. L.A. Nguyen and A. Szałas. An optimal tableau decision procedure for Converse-PDL. In N.-T. Nguyen, T.-D. Bui, E. Szczepanik, and N.-B. Nguyen, editors, Proceedings of KSE’2009, pages 207–214. IEEE Computer Society, 2009.
22. L.A. Nguyen and A. Szałas. A tableau calculus for regular grammar logics with converse. In Renate A. Schmidt, editor, Proceedings of CADE-22, volume 5663 of LNCS, pages 421–436. Springer-Verlag, 2009.
23. L.A. Nguyen and A. Szałas. Checking consistency of an ABox w.r.t. global assumptions in PDL. Fundamenta Informaticae, 102(1):97–113, 2010.
24. V.R. Pratt. A near-optimal method for reasoning about action. J. Comp. Syst. Sci., 20(2):231–254, 1980.
25. W. Rautenberg. Modal tableau calculi and interpolation. JPL, 12:403–423, 1983.
26. M.Y. Vardi. Reasoning about the past with two-way automata. In K.G. Larsen, S. Skyum, and G. Winskel, editors, Proceedings of ICALP’98, volume 1443 of LNCS, pages 628–641. Springer, 1998.

\footnote{Recall the example with \{p, (\sigma)[\sigma^{-}\neg p, (\sigma)((p_1 \lor q_1) \land \ldots \land (p_n \lor q_n))\}.}