An Optimal Tableau Decision Procedure for Converse-PDL *

Linh Anh Nguyen[1] and Andrzej Szałas[1,2]

[1] Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
{nguyen,andsz}@mimuw.edu.pl
[2] Dept. of Computer and Information Science, Linköping University
SE-581 83 Linköping, Sweden

Abstract. We give a novel tableau calculus and an optimal (ExpTime) tableau decision procedure based on the calculus for the satisfiability problem of propositional dynamic logic with converse. Our decision procedure is formulated with global caching and can be implemented together with useful optimization techniques.

Keywords: Converse-PDL, automated reasoning, tableaux

1 Introduction

Propositional dynamic logic (PDL) is widely used in areas such as program verification, theory of action and change, and knowledge representation (see, e.g., [13,2,4]). PDL can also be used as a syntactic variant of description logics and applied in reasoning about structured knowledge. The correspondence between PDL and description logics was first described by Schild [22]. By extending PDL with the converse operator, number restriction constructors, and "individuals" (for expressing ABoxes), one obtains the expressive description logic $\mathcal{CIQ}$ [7].

One of the basic inference problems in PDL depends on checking satisfiability of a set of formulas. Other inference problems in PDL are usually reducible to this problem. Fischer and Ladner [6] proved that the satisfiability problem of PDL is ExpTime-complete. It is then desirable to provide as efficient decision procedures as possible.

Efficient decision procedures for multimodal logics and description logics are often based on tableaux [20,1]. This approach is natural and allows for many useful optimization techniques [14,5,16]. The first ExpTime tableau decision procedure for PDL was given by Pratt [20]. His procedure is essentially based on constructing an "and-or" graph for the considered set of formulas by using tableau rules and global caching, and then checking whether a model for the set can be extracted from the graph. However, the formulation of his procedure is rather indirect: it goes via a labeled tableau calculus, tree-like labeled tableaux, tree-like traditional ("lean") tableaux, and "and-or" graphs.

By extending PDL with the converse operator we obtain the logic CPDL. This logic is more expressive than PDL [13, Theorem 10.15], but has the same complexity ExpTime for the satisfiability problem as PDL [24].[3] In [3], De Giacomo and Massacci gave

* This work is an extension of the conference paper [17]. It was supported by grants N N206 3982 33 and N N206 399334 from the Polish Ministry of Science and Higher Education.

[3] Using CPDL as a description logic, the converse operator is very useful: for example, the converse of role *has_child* means *has_parent*.

a non-optimal NExpTime tableau algorithm for checking satisfiability in CPDL and described how to transform the algorithm to an ExpTime version. However, the description is informal and unclear: the transformation is based on Pratt's global caching method formulated for PDL [20], but no global caching method has been formalized and proved sound for labeled tableaux that allow modifying labels of ancestor nodes in order to deal with converse. In [5], Donini and Massacci stated that the caching optimization technique *"prunes heavily the search space but its unrestricted usage may lead to unsoundness [37]. It is conjectured that 'caching' leads to* ExpTime*-bounds but this has not been formally proved so far, nor the correctness of caching has been shown"* [5, Page 89].[4] Furthermore, implementing a variation of the mentioned algorithm of De Giacomo and Massacci for PDL, Schmidt could not make her program run correctly for all test cases [23]. This at least means that the algorithm is not easy to implement. As far as we know and according to [1, page 26], no ExpTime tableau provers have been implemented for CPDL.

In this work we give an ExpTime tableau decision procedure for the satisfiability problem of CPDL, to which various useful optimization techniques can be applied. Our procedure for CPDL is based on Pratt's algorithm for PDL [20] and our tableau decision procedure for regular grammar logics with converse [18]. Similarly to [18], we deal with converse by using an analytic cut rule, which is a kind of "guessing the future" for nodes in traditional (unlabeled) tableaux. Our formulation of tableaux is directly based on constructing an "and-or" graph by using traditional (unlabeled) tableau rules and global caching and is therefore simpler and more direct for implementation than the one given by Pratt for PDL [20].

The rest of this paper is structured as follows. In Section 2 we recall basic definitions. Next, in Section 3, we develop a tableau calculus for CPDL. In Section 4 we prove soundness and completeness of our calculus. In Section 5 we present our ExpTime decision procedure for CPDL, based on the calculus. Section 6 is devoted to some optimizations, important for implementations. Finally, Section 7 concludes this work.

2 Propositional Dynamic Logic with Converse

We use Π_0 to denote the set of *atomic programs*, and Φ_0 to denote the set of *propositions* (also called *atomic formulas*). We denote elements of Π_0 by letters like σ. Elements of Φ_0 are denoted by letters like p and q.

Definition 2.1. A *Kripke model* is a pair $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$, where $\Delta^{\mathcal{M}}$ is a set of *states*, and $\cdot^{\mathcal{M}}$ is an interpretation function that maps each proposition $p \in \Phi_0$ to a subset $p^{\mathcal{M}}$ of $\Delta^{\mathcal{M}}$, and each atomic program $\sigma \in \Pi_0$ to a binary relation $\sigma^{\mathcal{M}}$ on $\Delta^{\mathcal{M}}$. ◁

Intuitively, for $p \in \Phi_0$, $p^{\mathcal{M}}$ is the set of states, where p is true and for $\sigma \in \Pi_0$, $\sigma^{\mathcal{M}}$ is interpreted as a binary relation consisting of pairs (input_state, output_state).

Definition 2.2. *Formulas* and *programs* of the *base language* of CPDL are defined respectively by the two following BNF grammar rules:

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \rightarrow \varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\alpha\rangle\varphi \mid [\alpha]\varphi$$
$$\alpha ::= \sigma \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \alpha^- \mid \varphi?$$ ◁

[4] Goré and Nguyen have recently formalized sound global caching [9,10,11,12] for traditional (unlabeled) tableaux in a number of modal logics without the * operator, which never look back at ancestor nodes.

We use letters like α, β to denote programs, and φ, ψ, ξ to denote formulas. The intended meaning of program operators is the following:

- $\alpha;\beta$ stands for a sequential composition of α and β,
- $\alpha \cup \beta$ stands for set-theoretical union of α and β,
- α^* stands for the reflexive and transitive closure of α,
- α^- stands for the converse of α,
- $\varphi?$ stands for the test operator.

Informally, a formula $\langle\alpha\rangle\varphi$ represents the set of states x such that the program α has a transition from x to a state y satisfying φ. Dually, a formula $[\alpha]\varphi$ represents the set of states x from which every transition of α leads to a state satisfying φ.

Formally, the interpretation function of a Kripke model $\mathcal{M}$ is extended to interpret complex formulas and complex programs as follows:

$$\top^{\mathcal{M}} \stackrel{\text{def}}{=} \Delta^{\mathcal{M}}, \quad \bot^{\mathcal{M}} \stackrel{\text{def}}{=} \emptyset$$
$$(\neg\varphi)^{\mathcal{M}} \stackrel{\text{def}}{=} \Delta^{\mathcal{M}} \setminus \varphi^{\mathcal{M}}, \quad (\varphi \to \psi)^{\mathcal{M}} \stackrel{\text{def}}{=} (\neg\varphi \vee \psi)^{\mathcal{M}}$$
$$(\varphi \wedge \psi)^{\mathcal{M}} \stackrel{\text{def}}{=} \varphi^{\mathcal{M}} \cap \psi^{\mathcal{M}}, \quad (\varphi \vee \psi)^{\mathcal{M}} \stackrel{\text{def}}{=} \varphi^{\mathcal{M}} \cup \psi^{\mathcal{M}}$$
$$(\langle\alpha\rangle\varphi)^{\mathcal{M}} \stackrel{\text{def}}{=} \{x \in \Delta^{\mathcal{M}} \mid \exists y(\, \alpha^{\mathcal{M}}(x,y) \wedge \varphi^{\mathcal{M}}(y)\,)\}$$
$$([\alpha]\varphi)^{\mathcal{M}} \stackrel{\text{def}}{=} \{x \in \Delta^{\mathcal{M}} \mid \forall y(\, \alpha^{\mathcal{M}}(x,y) \to \varphi^{\mathcal{M}}(y)\,)\}$$
$$(\alpha;\beta)^{\mathcal{M}} \stackrel{\text{def}}{=} \alpha^{\mathcal{M}} \circ \beta^{\mathcal{M}} = \{(x,y) \mid \exists z(\, \alpha^{\mathcal{M}}(x,z) \wedge \beta^{\mathcal{M}}(z,y)\,)\}$$
$$(\alpha \cup \beta)^{\mathcal{M}} \stackrel{\text{def}}{=} \alpha^{\mathcal{M}} \cup \beta^{\mathcal{M}}, \quad (\alpha^*)^{\mathcal{M}} \stackrel{\text{def}}{=} (\alpha^{\mathcal{M}})^*$$
$$(\alpha^-)^{\mathcal{M}} \stackrel{\text{def}}{=} (\alpha^{\mathcal{M}})^{-1} = \{(y,x) \mid (x,y) \in \alpha^{\mathcal{M}}\}, \quad (\varphi?)^{\mathcal{M}} \stackrel{\text{def}}{=} \{(x,x) \mid \varphi^{\mathcal{M}}(x)\}$$

Definition 2.3. Let $\mathcal{M}$ be a Kripke model and w be a state of $\mathcal{M}$. We write $\mathcal{M}, w \models \varphi$ to denote $w \in \varphi^{\mathcal{M}}$. For a set X of formulas, we write $\mathcal{M}, w \models X$ to denote that $\mathcal{M}, w \models \varphi$ for all $\varphi \in X$. If $\mathcal{M}, w \models \varphi$ (respectively $\mathcal{M}, w \models X$), then we say that $\mathcal{M}$ *satisfies* φ (respectively X) *at* w, and that φ (respectively X) is *satisfied at* w in $\mathcal{M}$. We say that $\mathcal{M}$ *validates* X if $\mathcal{M}, w \models X$ for all $w \in \Delta^{\mathcal{M}}$, and that X is *satisfiable* w.r.t. a set Γ of formulas used as *global assumptions* if there exists a Kripke model that validates Γ and satisfies X at some state. ◁

Definition 2.4. A formula or a program is in *negation-and-converse normal form* (NCNF) if it does not use the connective $\to$, uses the operator $\neg$ only immediately before propositions, and uses the converse program constructor $^-$ only for atomic programs. ◁

Proposition 2.5. *Every formula φ (respectively program α) can be transformed to a formula φ' (respectively a program α') in NCNF that is equivalent to φ (respectively α) in the sense that for every Kripke model $\mathcal{M}$, $\varphi^{\mathcal{M}} = (\varphi')^{\mathcal{M}}$ (respectively, $\alpha^{\mathcal{M}} = (\alpha')^{\mathcal{M}}$).*

Proof. Transform the considered formula (respectively program) by recursively replacing subformulas and subprograms according to Table 1 (this includes also replacements for subformulas occurring in tests). It is easy to see that this transformation satisfies the requirements. ◁

a subformula	$\psi \rightarrow \xi$	$\neg\top$	$\neg\bot$	$\neg\neg\psi$	$\neg(\psi \wedge \xi)$	$\neg(\psi \vee \xi)$	$\neg\langle\beta\rangle\psi$	$\neg[\beta]\psi$
to be replaced by	$\neg\psi \vee \xi$	$\bot$	$\top$	ψ	$\neg\psi \vee \neg\xi$	$\neg\psi \wedge \neg\xi$	$[\beta]\neg\psi$	$\langle\beta\rangle\neg\psi$

a program	$(\beta;\gamma)^-$	$(\beta \cup \gamma)^-$	$(\beta^*)^-$	$(\beta^-)^-$	$(\psi?)^-$
to be replaced by	$(\gamma^-;\beta^-)$	$\beta^- \cup \gamma^-$	$(\beta^-)^*$	β	$(\psi?)$

Table 1. Transformation of formulas and programs into negation-and-converse normal form

For example, the NCNF of formula $\neg[((\sigma_1 \cup \sigma_2);\sigma_3^*;(\neg\neg p)?)^-](q \vee \neg r)$ is formula

$$\langle p?;(\sigma_3{}^-)^*;(\sigma_1{}^- \cup \sigma_2{}^-)\rangle(\neg q \wedge r).$$

In this paper we assume that formulas and programs are represented in NCNF. We write $\overline{\varphi}$ to denote the NCNF of $\neg\varphi$.

Definition 2.6. By *simple programs*, denoted by letters like ς, we understand elements of $\Pi_1 \stackrel{\text{def}}{=} \{\sigma, \sigma^- \mid \sigma \in \Pi_0\}$. If $\varsigma = \sigma^-$ then $\varsigma^- \stackrel{\text{def}}{=} \sigma$. ◁

3 A Tableau Calculus for CPDL

Let X and Γ be finite sets of formulas in NCNF of the base language of CPDL (see Definition 2.2). In this section, we develop a tableau calculus called $\mathcal{C}_{\text{CPDL}}$ for the problem of checking whether X is satisfiable w.r.t. the set Γ of global assumptions.

Formulas used in our calculus are formed in the language that extends the base language with auxiliary modal operators $\Box_\varsigma$, where $\varsigma \in \Pi_1$ is a simple program. The operator $\Box_\varsigma$ has the same semantics as $[\varsigma]$, i.e. $(\Box_\varsigma\varphi)^{\mathcal{M}} \stackrel{\text{def}}{=} ([\varsigma]\varphi)^{\mathcal{M}}$, and behaves in our calculus as a "blocked version" of $[\varsigma]$.

CPDL has the finite model property: if X is satisfiable w.r.t. Γ then to construct a Kripke model that satisfies X and validates Γ, one needs to explore only formulas from a finite set $cls(X \cup \Gamma)$ – the closure of $X \cup \Gamma$. To define this set we use the Fischer-Ladner closure specified as follows.

Definition 3.1. The *Fischer-Ladner closure* $FL(\varphi)$ and the sets $FL^{\Box}([\alpha]\varphi)$ and $FL^{\Diamond}(\langle\alpha\rangle\varphi)$ for a formula φ and a program α in NCNF, without $\Box_\varsigma$, are the sets of formulas defined as follows:

$$FL(\top) \stackrel{\text{def}}{=} \{\top\},\ \ FL(\bot) \stackrel{\text{def}}{=} \{\bot\},\ \ FL(p) \stackrel{\text{def}}{=} \{p\},\ \ FL(\neg p) \stackrel{\text{def}}{=} \{\neg p\},$$
$$FL(\varphi \wedge \psi) \stackrel{\text{def}}{=} \{\varphi \wedge \psi\} \cup FL(\varphi) \cup FL(\psi),$$
$$FL(\varphi \vee \psi) \stackrel{\text{def}}{=} \{\varphi \vee \psi\} \cup FL(\varphi) \cup FL(\psi),$$
$$FL([\alpha]\varphi) \stackrel{\text{def}}{=} FL^{\Box}([\alpha]\varphi) \cup FL(\varphi),\ \ FL(\langle\alpha\rangle\varphi) \stackrel{\text{def}}{=} FL^{\Diamond}(\langle\alpha\rangle\varphi) \cup FL(\varphi),$$

$FL^{\Box}([\varsigma]\varphi) \stackrel{\text{def}}{=} \{[\varsigma]\varphi\}$,
$FL^{\Box}([\alpha;\beta]\varphi) \stackrel{\text{def}}{=} \{[\alpha;\beta]\varphi\} \cup FL^{\Box}([\alpha][\beta]\varphi) \cup FL^{\Box}([\beta])\varphi$,
$FL^{\Box}([\alpha \cup \beta]\varphi) \stackrel{\text{def}}{=} \{[\alpha \cup \beta]\varphi\} \cup FL^{\Box}([\alpha]\varphi) \cup FL^{\Box}([\beta])\varphi$,
$FL^{\Box}([\alpha^*]\varphi) \stackrel{\text{def}}{=} \{[\alpha^*]\varphi\} \cup FL^{\Box}([\alpha][\alpha^*]\varphi)$,
$FL^{\Box}([\psi?]\varphi) \stackrel{\text{def}}{=} \{[\psi?]\varphi\} \cup FL(\overline{\psi})$,

$FL^{\Diamond}(\langle\varsigma\rangle\varphi) \stackrel{\text{def}}{=} \{\langle\varsigma\rangle\varphi\}$,
$FL^{\Diamond}(\langle\alpha;\beta\rangle\varphi) \stackrel{\text{def}}{=} \{\langle\alpha;\beta\rangle\varphi\} \cup FL^{\Diamond}(\langle\alpha\rangle\langle\beta\rangle\varphi) \cup FL^{\Diamond}(\langle\beta\rangle)\varphi$,
$FL^{\Diamond}(\langle\alpha \cup \beta\rangle\varphi) \stackrel{\text{def}}{=} \{\langle\alpha \cup \beta\rangle\varphi\} \cup FL^{\Diamond}(\langle\alpha\rangle\varphi) \cup FL^{\Diamond}(\langle\beta\rangle)\varphi$,
$FL^{\Diamond}(\langle\alpha^*\rangle\varphi) \stackrel{\text{def}}{=} \{\langle\alpha^*\rangle\varphi\} \cup FL^{\Diamond}(\langle\alpha\rangle\langle\alpha^*\rangle\varphi)$,
$FL^{\Diamond}(\langle\psi?\rangle\varphi) \stackrel{\text{def}}{=} \{\langle\psi?\rangle\varphi\} \cup FL(\psi)$. ◁

Definition 3.2. For a set Y of formulas, define the *closure* $cls(Y)$ to be the set of formulas of the form φ and $\Box_\varsigma\varphi$, where

- $\varsigma \in \Pi_1$, ς occurs in Y, and
- $\varphi \in FL(\psi) \cup FL(\overline{\psi})$, for some formula ψ of the base language such that $\psi \in Y$ or $\Box_{\varsigma'}\psi \in Y$, for some ς'. ◁

We will define tableaux as "and-or" graphs. The *contents* of a *node* v of an "and-or" graph is a data structure consisting of two sets $\mathcal{L}(v)$ and $rfs(v)$ of formulas, where $\mathcal{L}(v)$ is called the *label* of v, and $rfs(v)$ is called *the set of formulas that have been reduced by a static rule after the last application of the transitional rule*, which will be clarified shortly.

Our calculus $\mathcal{C}_{\text{CPDL}}$ is specified by means of a finite set of tableau rules used to expand nodes of "and-or" graphs. A *tableau rule* is specified by 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.

Usually, a tableau rule is written downwards, with a set of formulas above the line as the *premise*, representing 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 "firm" and we ignore the word "possible". An "or"-rule has the meaning that, if the premise is satisfiable w.r.t. Γ then some of the possible conclusions are also satisfiable w.r.t. Γ. On the other hand, an "and"-rule has the meaning that, if the premise is satisfiable w.r.t. Γ then all of the conclusions are also satisfiable w.r.t. Γ (possibly in different states of the model under construction). Recall that, apart from the labels, there are also sets $rfs(\_)$ to be specified for the successor nodes.

We use Y to denote a set of formulas, and write Y, φ for $Y \cup \{\varphi\}$.

$$(\bot_0)\ \frac{Y, \bot}{\bot} \qquad (\bot)\ \frac{Y, p, \neg p}{\bot}$$

$$(\wedge)\ \frac{Y, \varphi \wedge \psi}{Y, \varphi, \psi} \qquad (\vee)\ \frac{Y, \varphi \vee \psi}{Y, \varphi \mid Y, \psi}$$

$$(\Box_;)\ \frac{Y, [\alpha;\beta]\varphi}{Y, [\alpha][\beta]\varphi} \qquad (\Diamond_;)\ \frac{Y, \langle\alpha;\beta\rangle\varphi}{Y, \langle\alpha\rangle\langle\beta\rangle\varphi}$$

$$(\Box_\cup)\ \frac{Y, [\alpha \cup \beta]\varphi}{Y, [\alpha]\varphi, [\beta]\varphi} \qquad (\Diamond_\cup)\ \frac{Y, \langle\alpha \cup \beta\rangle\varphi}{Y, \langle\alpha\rangle\varphi \mid Y, \langle\beta\rangle\varphi}$$

$$(\Box_?)\ \frac{Y, [\psi?]\varphi}{Y, \overline{\psi} \mid Y, \varphi} \qquad (\Diamond_?)\ \frac{Y, \langle\psi?\rangle\varphi}{Y, \psi, \varphi}$$

$$(\Box_*)\ \frac{Y, [\alpha^*]\varphi}{Y, \varphi, [\alpha][\alpha^*]\varphi} \qquad (\Diamond_*)\ \frac{Y, \langle\alpha^*\rangle\varphi}{Y, \varphi \mid Y, \langle\alpha\rangle\langle\alpha^*\rangle\varphi}$$

$$(cut)\ \frac{Y}{Y, \varphi \mid Y, \Box_\varsigma\langle\varsigma^-\rangle\overline{\varphi}}\ \text{if (*)}$$

(*) : Y contains neither φ nor $\Box_\varsigma\langle\varsigma^-\rangle\overline{\varphi}$ but $\langle\varsigma\rangle\psi$ for some ψ, $[\varsigma^-]\varphi \in cls(Y \cup \Gamma)$, and $\varphi \notin rfs(v)$, where v is the current node to which the rule is applied

$$(trans)\ \frac{Y}{\&\{\ (\{\varphi\} \cup \{\psi \text{ s.t. } [\varsigma]\psi \in Y \text{ or } \Box_\varsigma\psi \in Y\} \cup \Gamma) \text{ s.t. } \langle\varsigma\rangle\varphi \in Y\ \}}$$

Fig. 1. Rules of the tableau calculus $\mathcal{C}_{\mathrm{CPDL}}$.

Definition 3.3. Define *tableau calculus* $\mathcal{C}_{\mathrm{CPDL}}$ *w.r.t. a set* Γ *of global assumptions* to be the set of the tableau rules given in Figure 1. The rule (*trans*) is the only "and"-rule and the only *transitional rule*. The other rules of $\mathcal{C}_{\mathrm{CPDL}}$ are "or"-rules, which are also called *static rules*.[5]

For any rule of $\mathcal{C}_{\mathrm{CPDL}}$ except (*cut*) and (*trans*), the distinguished formulas of the premise are called the *principal formulas* of the rule.[6] The principal formulas of the rule (*trans*) are the formulas of the form $\langle\sigma\rangle\varphi$ of the premise. The rule (*cut*) does not have principal formulas.

We assume that any of the rules $(\wedge)$, $(\vee)$, $(\Box_;)$, $(\Box_\cup)$, $(\Box_?)$, $(\Box_*)$ is applicable to a node v only when the principal formula does not belong to $rfs(v)$.

Applying a static rule different from $(\bot_0)$ and $(\bot)$ to a node v, for any successor node w of v, we set $rfs(w)$ to be the set that extends $rfs(v)$ with the principal formula of the applied rule. Applying any other rule to a node v, for any successor node w of v, we set $rfs(w) = \emptyset$. ◁

Instantiating, for example, the rule (*trans*) to $Y = \{\langle\sigma\rangle p, \langle\sigma\rangle q, [\sigma]r\}$ and $\Gamma = \{s\}$ we get two conclusions: $\{p, r, s\}$ and $\{q, r, s\}$.

[5] Unary static rules can be treated either as "and"-rules or as "or"-rules.
[6] I.e. principal formulas are the ones not belonging to Y.

Remark 3.4. The intuition of distinguishing between static and transitional is that static rules keep us in the same state of the model under construction, while each conclusion of the transitional rule takes us to a new state. ◁

Recall that $rfs(w)$ stands for "the set of formulas that have been reduced by a static rule after the last application of the transitional rule". By using $rfs(\_)$ and the restriction on applicability of the rules $(\wedge)$, $(\vee)$, $(\Box_;)$, $(\Box_\cup)$, $(\Box_?)$, $(\Box_*)$, and (cut), in a sequence of applications of static rules a formula of the form $\varphi \wedge \psi$, $\varphi \vee \psi$, $[\alpha;\beta]\varphi$, $[\alpha \cup \beta]\varphi$, $[\psi?]\varphi$, or $[\alpha^*]\varphi$ may be reduced (as a principal formula) at most once. We do not adopt the restriction for the rules $(\Diamond_;)$, $(\Diamond_\cup)$, $(\Diamond_?)$, and $(\Diamond_*)$ because we will require formulas of the form $\langle\alpha\rangle\varphi$ to be "realized" (in a finite number of steps).

We assume the following *preferences for the rules of* $\mathcal{C}_{\mathrm{CPDL}}$: the rules $(\bot_0)$ and $(\bot)$ have the highest priority; unary static rules have a higher priority than non-unary static rules; all the static rules have a higher priority than the transitional rule $(trans)$.

Definition 3.5. An *"and-or" graph for* (X, Γ), also called a *tableau for* (X, Γ), is an "and-or" graph defined as follows:

- the initial node ν of the graph, called the *root* of the graph, is specified by $\mathcal{L}(\nu) = X \cup \Gamma$ and $rfs(\nu) = \emptyset$,
- for every node v of the graph, if a tableau rule of $\mathcal{C}_{\mathrm{CPDL}}$ is applicable to the label of v in the sense that an instance of the rule has $\mathcal{L}(v)$ as the premise and $Z_1, \ldots, Z_k$ as the possible conclusions, then choose such a rule accordingly to the preferences[7] and apply it to v to create k successors $w_1, \ldots, w_k$ of v with $\mathcal{L}(w_i) = Z_i$ for $1 \leq i \leq k$, maintaining the following constraints:
  - if the graph already contains a node w'_i with the same contents as w_i then instead of creating a new node w_i as a successor of v we just connect v to w'_i and assume $w_i = w'_i$,
  - if the applied rule is $(trans)$ then we *label* the edge (v, w_i) by the principal formula corresponding to the successor w_i.

If the rule expanding v is an "or"-rule then v is an *"or"-node*, else v is an *"and"-node*. If no rule is applicable to v then v is an *end node*. ◁

Note that each node of the graph is "expanded" only once (using one rule), the graph is constructed using *global caching* [20,10,12] and the contents of its nodes are unique, and the sets $\mathcal{L}(v)$ and $rfs(v)$ of each node v consist of formulas from $cls(X \cup \Gamma)$.

Notice the restrictions on applicability of the rule (cut). Observe that if X and Γ do not contain the converse constructor then the rule (cut) is not used for the construction of any "and-or" graph for (X, Γ). In that case, the operators $\Box_\varsigma$ will not appear, and we can simplify the rule $(trans)$ appropriately and delete the rule (cut) to obtain a simpler calculus for PDL (without converse).

Definition 3.6. A *marking* of an "and-or" graph G is a subgraph G' of G such that:

- the root of G is the root of G',

[7] If there are several applicable rules with the same priority, choose any of them.

- if v is a node of G' and is an "or"-node of G then there exists at least one edge (v, w) of G that is an edge of G',
- if v is a node of G' and is an "and"-node of G then every edge (v, w) of G is an edge of G',
- if (v, w) is an edge of G' then v and w are nodes of G'. ◁

Definition 3.7. Let G be an "and-or" graph for (X, Γ), G' be a marking of G, v be a node of G', and $\langle\alpha\rangle\varphi$ be a formula of the label of v. A *trace* of $\langle\alpha\rangle\varphi$ in G' starting from v is a sequence $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ such that:

- $v_0 = v$ and $\varphi_0 = \langle\alpha\rangle\varphi$,
- for every $1 \leq i \leq k$, (v_{i-1}, v_i) is an edge of G',
- for every $1 \leq i \leq k$, φ_i is a formula of the label of v_i such that:
  - if φ_{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 $(\Diamond_;)$, $(\Diamond_\cup)$ or $(\Diamond_*)$ then φ_i is the formula obtained from φ_{i-1},
  - else if the rule is $(\Diamond_?)$ and $\varphi_{i-1} = \langle\psi?\rangle\xi$ then $\varphi_i = \xi$,
  - else the rule is $(trans)$, φ_{i-1} is of the form $\langle\sigma\rangle\xi$ and is the label of the edge (v_{i-1}, v_i), and $\varphi_i = \xi$. ◁

Definition 3.8. A trace $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ of $\langle\alpha\rangle\varphi$ in G' is called a $\Diamond$-*realization* in G' for $\langle\alpha\rangle\varphi$ at v_0 if $\varphi_k = \varphi$. ◁

Definition 3.9. A marking G' of an "and-or" graph G for (X, Γ) is *consistent* if:

- *local consistency:* G' does not contain any node with label $\{\bot\}$,
- *global consistency:* for every node v of G', every formula of the form $\langle\alpha\rangle\varphi$ of the label of v has a $\Diamond$-realization (starting at v) in G'. ◁

In Figure 2 we give a part of an "and-or" graph for $(\{\neg p, \langle\sigma^*\rangle[(\sigma^-)^*]p\}, \emptyset)$. The formula $\langle(\sigma^-)^*\rangle\neg p$ at the node (13) does not have any $\Diamond$-realization in any marking with the local consistency property. Hence, the nodes (10), (12)-(16), (18) do not belong to any potential consistent marking. Consequently, the formula $\langle\sigma\rangle\langle\sigma^*\rangle[(\sigma^-)^*]p$ of the node (17) does not have any $\Diamond$-realization in any potential consistent marking. Therefore, without expanding the nodes (19) and (20), one can conclude that the graph does not have any consistent marking. By Theorem 3.10 given below, it follows that the set $\{\neg p, \langle\sigma^*\rangle[(\sigma^-)^*]p\}$ is unsatisfiable (w.r.t. $\emptyset$).

Theorem 3.10 (Soundness and Completeness). *Let X and Γ be finite sets of formulas in NCNF of the base language, and G be an "and-or" graph for (X, Γ). Then X is satisfiable w.r.t. the set Γ of global assumptions iff G has a consistent marking.* ◁

The "only if" direction means soundness of $\mathcal{C}_{\mathrm{CPDL}}$, while the "if" direction means completeness of $\mathcal{C}_{\mathrm{CPDL}}$. This theorem follows from Lemmas 4.1 and 4.12, which are given and proved in the next section.

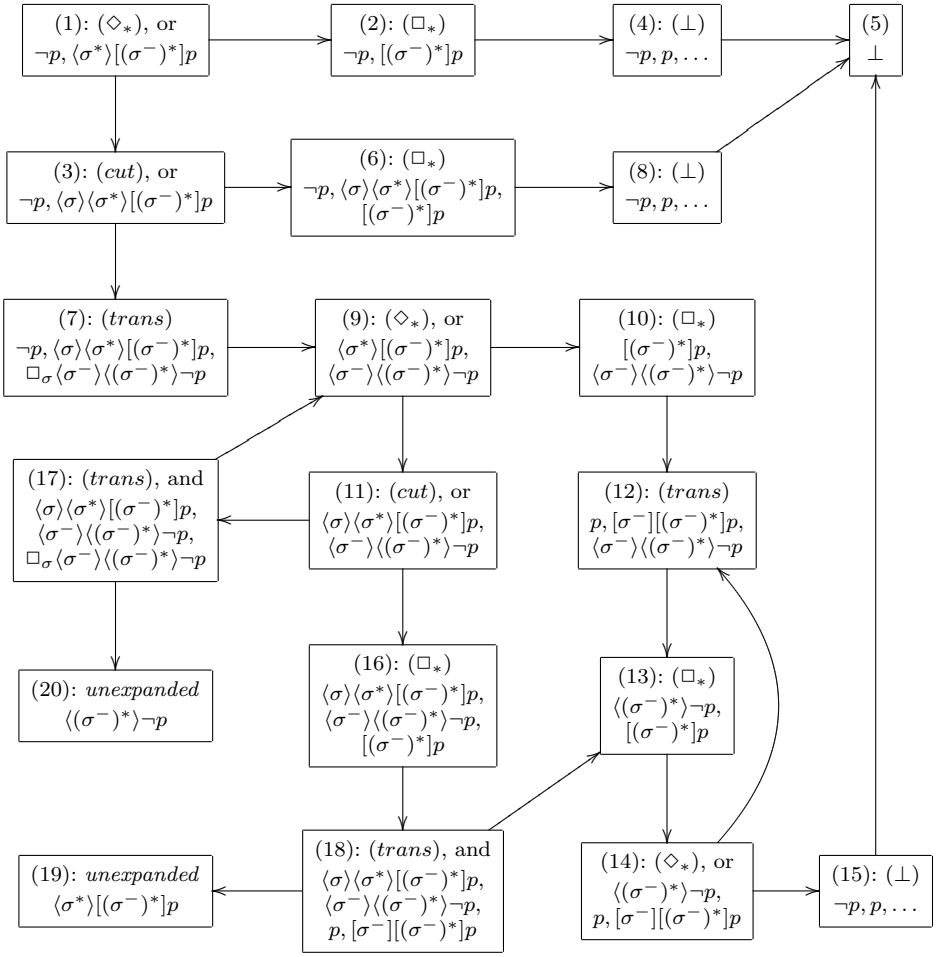

Fig. 2. A part of an "and-or" graph for $(\{\neg p, \langle\sigma^*\rangle[(\sigma^-)^*]p\}, \emptyset)$. In the main part of each node we display the formulas of the label of the node. We do not display the sets $rfs(\_)$ of the nodes. The edges $((7),(9))$, $((17),(9))$, and $((18),(19))$ are labeled by $\langle\sigma\rangle\langle\sigma^*\rangle[(\sigma^-)^*]p$. The edges $((12),(13))$, $((18),(13))$, and $((17),(20))$ are labeled by $\langle\sigma^-\rangle\langle(\sigma^-)^*\rangle\neg p$. To derive inconsistency of the graph, one does not need to expand the nodes (19) and (20).

4 Proofs of Soundness and Completeness

The alphabet $\Sigma(\alpha)$ of a program α is defined as follows: $\Sigma(\varsigma) = \{\varsigma\}$, $\Sigma(\varphi?) = \{\varphi?\}$, $\Sigma(\beta;\gamma) = \Sigma(\beta) \cup \Sigma(\gamma)$, $\Sigma(\beta \cup \gamma) = \Sigma(\beta) \cup \Sigma(\gamma)$, $\Sigma(\beta^*) = \Sigma(\beta)$. Thus, $\Sigma(\alpha)$ contains not only atomic programs but also expressions of the form σ^- or $(\varphi?)$.

A program α is a regular expression over its alphabet $\Sigma(\alpha)$. The regular language $\mathcal{L}(\alpha)$ generated by α is specified as follows: $\mathcal{L}(\varsigma) = \{\varsigma\}$, $\mathcal{L}(\varphi?) = \{\varphi?\}$, $\mathcal{L}(\beta \cup \gamma) = \mathcal{L}(\beta) \cup \mathcal{L}(\gamma)$, $\mathcal{L}(\beta;\gamma) = \mathcal{L}(\beta).\mathcal{L}(\gamma)$, and $\mathcal{L}(\beta^*) = (\mathcal{L}(\beta))^*$, where if L and M are sets of words then $L.M = \{\alpha\beta \mid \alpha \in L, \beta \in M\}$ and $L^* = \bigcup_{n\geq 0} L^n$ with $L^0 = \{\varepsilon\}$ and $L^{n+1} = L.L^n$, where ε is the empty word. We treat words of $\mathcal{L}(\alpha)$ also as programs, e.g. $\sigma_1(\varphi?)\sigma_2{}^-$ as $(\sigma_1;\varphi?;\sigma_2{}^-)$.

By $\mathcal{G}(\alpha)$ we denote the context-free grammar over alphabet $\Sigma(\alpha)$, which is specified as follows: the grammar variables are sub-expressions of α that do not belong to $\Sigma(\alpha)$, the starting symbol is α, and the grammar rules are:

$$(\beta;\gamma) ::= \beta\gamma \qquad (\beta \cup \gamma) ::= \beta \mid \gamma \qquad (\beta^*) ::= \varepsilon \mid \beta(\beta^*)$$

Given a modality $\triangle$ of the form $\langle\alpha_1\rangle \dots \langle\alpha_k\rangle$ or $[\alpha_1] \dots [\alpha_k]$ we call $\alpha_1 \dots \alpha_k$ the *program sequence corresponding to* $\triangle$.

4.1 Soundness

Lemma 4.1 (Soundness). *Let X and Γ be finite sets of formulas in NCNF of the base language, and G be an "and-or" graph for (X, Γ). Suppose that X is satisfiable w.r.t. the set Γ of global assumptions. Then G has a consistent marking.*

Proof. We construct a consistent marking G' of G as follows. At the beginning, G' contains only the root of G. Then, for every node v of G' and for every successor w of v in G, if the label of w is satisfiable w.r.t. Γ, then add the node w and the edge (v, w) to G'.

To prove that G' is a marking of G we need to show that:

i) for every "or"-rule of $\mathcal{C}_{\mathrm{CPDL}}$, if the premise is satisfiable w.r.t. Γ then one of the possible conclusions of the rule is also satisfiable w.r.t. Γ,
ii) for every "and"-rule of $\mathcal{C}_{\mathrm{CPDL}}$, if the premise is satisfiable w.r.t. Γ then so is each conclusion of the rule.

We consider here only the rule (*cut*) and leave the others to the reader. Suppose that Y is satisfiable w.r.t. Γ and let $\mathcal{M}$ be a model that validates Γ and satisfies Y at a state u. Suppose that $\mathcal{M}, u \not\models \Box_\varsigma\langle\varsigma^-\rangle\overline{\varphi}$. We show that $\mathcal{M}, u \models \varphi$. We have that $\mathcal{M}, u \models \langle\varsigma\rangle[\varsigma^-]\varphi$. Hence there exists u' such that $(u, u') \in \varsigma^{\mathcal{M}}$ and $\mathcal{M}, u' \models [\varsigma^-]\varphi$. It follows that $(u', u) \in (\varsigma^-)^{\mathcal{M}}$ and $\mathcal{M}, u \models \varphi$.

Clearly, G' satisfies the local consistency property.

We now check the global consistency property of G'. Let v_0 be a node of G', Y be the label of v_0, and $\langle\alpha\rangle\varphi$ be a formula of Y. We show that the formula has a $\Diamond$-realization (starting from v_0) in G'. As Y is satisfiable w.r.t. Γ, there exists a Kripke model $\mathcal{M}$ that validates Γ and satisfies Y at a state u_0. Since $\langle\alpha\rangle\varphi$ is satisfied at u_0 in $\mathcal{M}$, there exist a word $\delta = \varsigma_1 \dots \varsigma_{j_1}(\psi_1?)\varsigma_{j_1+1} \dots \varsigma_{j_2}(\psi_2?) \dots \varsigma_{j_k} \in \mathcal{L}(\alpha)$ (with $0 \le j_1 \le j_2 \le \dots \le j_k$) and states $u_1, \dots, u_{j_k}$ of $\mathcal{M}$ such that: $(u_{j-1}, u_j) \in \varsigma_j^{\mathcal{M}}$ for $1 \le j \le j_k$, $u_{j_l} \in \psi_l^{\mathcal{M}}$ for $1 \le l \le k-1$, and $u_{j_k} \in \varphi^{\mathcal{M}}$. Denote this property by $(\star)$.

We construct a $\Diamond$-realization $(v_0, \varphi_0), \dots, (v_h, \varphi_h)$ in G' for $\langle\alpha\rangle\varphi$ at v_0 and a map $f : \{v_0, \dots, v_h\} \to \{u_0, \dots, u_{j_k}\}$ such that $f(v_0) = u_0$, $f(v_h) = u_{j_k}$, and for every $0 \le i < h$, if $f(v_i) = u_j$ then $f(v_{i+1})$ is either u_j or u_{j+1}. For $1 \le i \le h$, let $\triangle_i$ be the sequence of existential modal operators such that $\varphi_i = \triangle_i\varphi$ and let S_i be the program sequence corresponding to $\triangle_i$. We maintain the following invariants for $0 \le i \le h$:

(a) The chain $(v_0, \varphi_0), \dots, (v_i, \varphi_i)$ is a trace of $\langle\alpha\rangle\varphi$ in G'.
(b) The label of v_i is satisfied at the state $f(v_i)$ of $\mathcal{M}$.
(c) If $f(v_i) = u_j$ and $j \notin \{j_1, \dots, j_{k-1}\}$ then the suffix δ_i of δ that starts from ς_{j+1} is derivable from S_i using a left derivation of the context-free grammar $\mathcal{G}(\alpha)$.

(d) If $f(v_i) = u_j$ and $j_{l-1} < j = j_l = j_{l+1} = \ldots = j_{l+m} < j_{l+m+1}$ then there exists $0 \leq n \leq m+1$ such that the suffix δ_i of δ that starts from $(\psi_{l+n}?)$ if $n \leq m$ or from ς_{j+1} if $n = m+1$ is derivable from S_i using a left derivation of the context-free grammar $\mathcal{G}(\alpha)$.

With $\varphi_0 = \langle\alpha\rangle\varphi$ and $f(v_0) = u_0$, the invariants clearly hold for $i = 0$.
Set $i := 0$. While $\varphi_i \neq \varphi$ do:

- Case v_i is expanded using a static rule and φ_i is the principal formula:
  - Case $\varphi_i = \langle\beta;\gamma\rangle\psi$: Let v_{i+1} be the only successor of v_i, $\varphi_{i+1} = \langle\beta\rangle\langle\gamma\rangle\psi$, $f(v_{i+1}) = f(v_i)$, and set $i := i+1$. Clearly, the invariants still hold.
  - Case $\varphi_i = \langle\xi?\rangle\psi$: Let v_{i+1} be the only successor of v_i, $\varphi_{i+1} = \psi$, and $f(v_{i+1}) = f(v_i)$. Observe that the invariant (a) clearly holds for $i+1$. As φ_i is satisfied at $f(v_i)$, both ξ and ψ are satisfied at $f(v_{i+1}) = f(v_i)$. Hence, the label of v_{i+1} is satisfied at $f(v_{i+1})$, and the invariant (b) holds for $i+1$. Let $f(v_i) = u_j$. By the invariants (c) and (d) for i, we have that $j \in \{j_1, \ldots, j_{k-1}\}$. As δ_i is derivable from $S_i = (\xi?)S_{i+1}$ using a left derivation of the context-free grammar $\mathcal{G}(\alpha)$, the word δ_{i+1} such that $\delta_i = (\xi?)\delta_{i+1}$ is derivable from S_{i+1} using a left derivation of $\mathcal{G}(\alpha)$. Therefore, by setting $i := i+1$, the invariants (a)-(d) still hold (for the new i).
  - Case $\varphi_i = \langle\beta \cup \gamma\rangle\psi$: Let $\psi = \triangle'_i\varphi$ and let S'_i be the program sequence corresponding to $\triangle'_i$. By the invariants (c) and (d), δ_i is derivable from $(\beta\cup\gamma)S'_i$ using a left derivation of $\mathcal{G}(\alpha)$. If the first step of that derivation gives $\beta S'_i$ then let $\varphi_{i+1} = \langle\beta\rangle\psi$ else let $\varphi_{i+1} = \langle\gamma\rangle\psi$. By $(\star)$, it follows that φ_{i+1} is satisfied at the state $f(v_i)$. Let v_{i+1} be the successor of v_i such that φ_{i+1} belongs to the label of v_{i+1}. Clearly, the invariant (a) holds for $i+1$. Let $f(v_{i+1}) = f(v_i)$. Thus, the invariants (b)-(d) also hold for $i+1$. Therefore, by setting $i := i+1$, the invariants (a)-(d) still hold (for the new i).
  - Case $\varphi_i = \langle\beta^*\rangle\psi$: Let $\psi = \triangle'_i\varphi$ and let S'_i be the program sequence corresponding to $\triangle'_i$. By the invariants (c) and (d), δ_i is derivable from $(\beta^*)S'_i$ using a left derivation of $\mathcal{G}(\alpha)$. If the first step of that derivation gives S'_i then let $\varphi_{i+1} = \psi$ else let $\varphi_{i+1} = \langle\beta\rangle\langle\beta^*\rangle\psi$. Let v_{i+1} be the successor of v_i such that φ_{i+1} belongs to the label of v_{i+1}, let $f(v_{i+1}) = f(v_i)$, and set $i := i+1$. Similarly to the above case, the invariants (a)-(d) still hold.
- Case v_i is expanded using a static rule but φ_i is not the principal formula:
  - Case the principal formula is not of the form $\langle\alpha'\rangle\varphi'$: Let v_{i+1} be the successor of v_i such that (v_i, v_{i+1}) is an edge of G' and the label of v_{i+1} is satisfied at the state $f(v_i)$ of $\mathcal{M}$. Such a node v_{i+1} exists because the label of v_i is satisfied at the state $f(v_i)$ of $\mathcal{M}$. Let $\varphi_{i+1} = \varphi_i$, $f(v_{i+1}) = f(v_i)$, and set $i := i+1$. Clearly, the invariants still hold.
  - Case the principal formula is of the form $\langle\alpha'\rangle\varphi'$: During a sequence of applications of static rules between two applications of the transitional rule, proceed as for realizing $\langle\alpha'\rangle\varphi'$ in G' (like for the current $\Diamond$-realization of $\langle\alpha\rangle\varphi$ in G' at v_0). This decides how to choose v_{i+1} and has effects on terminating the trace (to obtain a $\Diamond$-realization for $\langle\alpha\rangle\varphi$ in G' at v_0). We also choose $\varphi_{i+1} = \varphi_i$ and $f(v_{i+1}) = f(v_i)$. By setting $i := i+1$, the invariants still hold (for the new i).

– Case v_i is expanded using the transitional rule: Let $f(v_i) = u_j$. Then, by the invariants (c) and (d), φ_i must be of the form $\langle \varsigma_{j+1} \rangle \psi$. Let (v_i, v_{i+1}) be the edge of G with the label φ_i. Let $\varphi_{i+1} = \psi$ and $f(v_{i+1}) = u_{j+1}$. Clearly, the invariant (a) holds for $i+1$. By $(\star)$, ψ is satisfied at the state u_{j+1} of $\mathcal{M}$. By the invariant (b), the other formulas of the label of v_{i+1} are also satisfied at the state u_{j+1} of $\mathcal{M}$. That is, the invariant (b) holds for $i+1$. It is easy to see that the invariants (c) and (d) remain true after increasing i by 1. So, by setting $i := i+1$, all the invariants (a)-(d) still hold.

It remains to 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 $\langle \alpha \rangle \varphi$ in G' eventually ends because: each formula of the form $\psi \wedge \xi$, $\psi \vee \xi$, or $[\beta]\psi$ with $\beta \notin \Pi_1$ may be reduced at most once; each formula of the form $\langle \beta \rangle \psi$ with $\beta \notin \Pi_1$ of the label of any node among $v_0, \ldots, v_i$ is reduced according to some $\Diamond$-realization.

Therefore, 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 $f(v_i) = u_j$ then $f(v_{i+1}) = u_{j+1}$. As the image of f is $\{u_0, \ldots, u_{j_k}\}$, the construction of the trace must end at some step (with $\varphi_i = \varphi$) and we obtain a $\Diamond$-realization in G' for $\langle \alpha \rangle \varphi$ at v_0. This completes the proof. ◁

4.2 Model Graphs

We will prove completeness of $\mathcal{C}_{\mathrm{CPDL}}$ via model graphs. The technique has previously been used in [21,8,15,18].

Definition 4.2. A *model graph* is a tuple $\langle W, (R_\varsigma)_{\varsigma \in \Pi_1}, H \rangle$, where W is a set of nodes, R_ς for $\varsigma \in \Pi_1$ is a binary relation on W, and H is a function that maps each node of W to a set of formulas. ◁

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

A trace of a formula $\langle \alpha \rangle \varphi$ at a node in a model graph is defined analogously as for the case of "and-or" graphs:

Definition 4.3. Given a model graph $\mathcal{M} = \langle W, (R_\varsigma)_{\varsigma \in \Pi_1}, H \rangle$ and a node $v \in W$, a *trace* of a formula $\langle \alpha \rangle \varphi \in H(v)$ (starting from v) is a chain $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ such that:

– $v_0 = v$ and $\varphi_0 = \langle \alpha \rangle \varphi$,
– 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:
  • if $\varphi_{i-1} = \langle \beta; \gamma \rangle \psi$ then $\varphi_i = \langle \beta \rangle \langle \gamma \rangle \psi$,
  • else if $\varphi_{i-1} = \langle \beta \cup \gamma \rangle \psi$ then $\varphi_i = \langle \beta \rangle \psi$ or $\varphi_i = \langle \gamma \rangle \psi$,
  • else if $\varphi_{i-1} = \langle \beta^* \rangle \psi$ then $\varphi_i = \psi$ or $\varphi_i = \langle \beta \rangle \langle \beta^* \rangle \psi$,
  • else $\varphi_{i-1} = \langle \psi? \rangle \xi$ and $\varphi_i = \xi$,
– for every $1 \leq i \leq k$, if $v_i \neq v_{i-1}$ then:

- φ_{i-1} is of the form $\langle\varsigma\rangle\psi$ and $\varphi_i = \psi$ and $(v_{i-1}, v_i) \in R_\varsigma$. ◁

Definition 4.4. A trace (v_0, φ_0), ..., (v_k, φ_k) of $\langle\alpha\rangle\varphi$ in a model graph $\mathcal{M}$ is called a *$\Diamond$-realization* for $\langle\alpha\rangle\varphi$ at v_0 if $\varphi_k = \varphi$. ◁

Similarly as for markings of "and-or" graphs, we define that:

Definition 4.5. A model graph $\mathcal{M} = \langle W, (R_\varsigma)_{\varsigma\in\Pi_1}, H\rangle$ is *consistent* if:

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

◁

Definition 4.6. A model graph $\mathcal{M} = \langle W, (R_\varsigma)_{\varsigma\in\Pi_1}, H\rangle$ is said to be *saturated* if the following conditions hold for every $v \in W$:

- for every $\varphi \in H(v)$:
  - if $\varphi = \psi \wedge \xi$ then $\{\psi, \xi\} \subset H(v)$,
  - if $\varphi = \psi \vee \xi$ then $\psi \in H(v)$ or $\xi \in H(v)$,
  - if $\varphi = \langle\psi?\rangle\xi$ then $\psi \in H(v)$,[8]
  - if $\varphi = [\alpha;\beta]\psi$ then $[\alpha][\beta]\psi \in H(v)$,
  - if $\varphi = [\alpha \cup \beta]\psi$ then $\{[\alpha]\psi, [\beta]\psi\} \subset H(v)$,
  - if $\varphi = [\psi?]\xi$ then $\overline{\psi} \in H(v)$ or $\xi \in H(v)$,
  - if $\varphi = [\alpha^*]\psi$ then $\{\psi, [\alpha][\alpha^*]\psi\} \subset H(v)$,
  - if $\varphi = [\varsigma]\psi$ and $R_\varsigma(v, w)$ holds then $\psi \in H(w)$,
  - if $\varphi = \Box_\varsigma\psi$ and $R_\varsigma(v, w)$ holds then $\psi \in H(w)$;
- if $R_\varsigma(v, w)$ holds and $[\varsigma^-]\varphi \in H(w)$ then $\varphi \in H(v)$ or $\Box_\varsigma\langle\varsigma^-\rangle\overline{\varphi} \in H(v)$. ◁

The last condition corresponds to the rule (*cut*). As shown in the proof of Lemma 4.9, it can be strengthened to "if $R_\varsigma(v, w)$ holds and $[\varsigma^-]\varphi \in H(w)$ then $\varphi \in H(v)$".

Definition 4.7. Given a model graph $\mathcal{M} = \langle W, (R_\varsigma)_{\varsigma\in\Pi_1}, H\rangle$, the Kripke model $\mathcal{M}'$ defined by $\Delta^{\mathcal{M}'} = W$, $\sigma^{\mathcal{M}'} = R_\sigma \cup (R_{\sigma^-})^{-1}$ for $\sigma \in \Pi_0$, and $p^{\mathcal{M}'} = \{w \in W \mid p \in H(w)\}$ for $p \in \Phi_0$ is called the *Kripke model corresponding to* $\mathcal{M}$. ◁

Define the NCNF of $\neg\Box_\varsigma\varphi$ to be $\langle\varsigma\rangle\overline{\varphi}$. Recall that the NCNF of $\neg[\varsigma]\varphi$ and $\neg\langle\varsigma\rangle\varphi$ are $\langle\varsigma\rangle\overline{\varphi}$ and $[\varsigma]\overline{\varphi}$, respectively.

Lemma 4.8. *Let $\mathcal{M} = \langle W, (R_\varsigma)_{\varsigma\in\Pi_1}, H\rangle$ be a consistent and saturated model graph and let $\mathcal{M}'$ be the Kripke model corresponding to $\mathcal{M}$. Then, for any $w \in W$, if $\mathcal{M}', w \models \varphi$ then $H(w)$ does not contain $\overline{\varphi}$.*

Proof. By induction on the structure of φ, using the global consistency. ◁

Lemma 4.9. *Let X and Γ be finite sets of formulas in NCNF of the base language and let $\mathcal{M} = \langle W, (R_\varsigma)_{\varsigma\in\Pi_1}, H\rangle$ be a consistent and 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 Kripke model $\mathcal{M}'$ corresponding to $\mathcal{M}$ validates Γ and satisfies X at τ.*

[8] The condition $\xi \in H(v)$ is taken care of by global consistency.

Proof. We first show that if $R_\varsigma(v,w)$ holds and $[\varsigma^-]\varphi \in H(w)$ then $\varphi \in H(v)$. For contradiction, suppose that $R_\varsigma(v,w)$ holds, $[\varsigma^-]\varphi \in H(w)$, but $\varphi \notin H(v)$. Since $\mathcal{M}$ is saturated, we have that $\Box_\varsigma\langle\varsigma^-\rangle\overline{\varphi} \in H(v)$. It follows that $\langle\varsigma^-\rangle\overline{\varphi} \in H(w)$, since $R_\varsigma(v,w)$ holds. By Lemma 4.8, this contradicts the assumption that $[\varsigma^-]\varphi \in H(w)$.

We prove by induction on the construction of φ that if $\varphi \in H(w_0)$ for an arbitrary $w_0 \in W$ then $\mathcal{M}', w_0 \models \varphi$. It suffices to consider only the non-trivial cases when φ is of the form $\langle\alpha\rangle\psi$ or $[\alpha]\psi$. Suppose that $\varphi \in H(w_0)$.

Consider the case $\varphi = \langle\alpha\rangle\psi$. Let $(w_0,\varphi_0),\ldots,(w_k,\varphi_k)$ be a $\Diamond$-realization for φ at w_0. We have that $\varphi_0 = \varphi$ and $\varphi_k = \psi$. Let $0 \le i_1 < \ldots < i_h < k$ be all the indices such that, for $1 \le j \le h$, φ_{i_j} is of the form $\langle\omega_{i_j}\rangle\varphi_{i_j+1}$ with ω_{i_j} of the form ς_{i_j} or $(\psi_{i_j}?)$. Observe that $\omega_{i_1}\omega_{i_2}\ldots\omega_{i_h} \in \mathcal{L}(\alpha)$ and there is a path from w_0 to w_k in $\mathcal{M}$ whose edges are sequentially labeled by those ω_{i_j} of the form ς_{i_j}. Since $\mathcal{M}$ is saturated, for $1 \le j \le h$, if $\omega_{i_j} = (\psi_{i_j}?)$ then $\psi_{i_j} \in H(w_{i_j})$, which, by the inductive assumption, implies that $\mathcal{M}', w_{i_j} \models \psi_j$. It follows that $(w_0,w_k) \in \alpha^{\mathcal{M}'}$. Since $\psi \in H(w_k)$, by the inductive assumption, we have $\mathcal{M}', w_k \models \psi$. Therefore $\mathcal{M}', w_0 \models \langle\alpha\rangle\psi$.

Consider now the case $\varphi = [\alpha]\psi$.

Let w be an arbitrary node of $\mathcal{M}$ such that $(w_0,w) \in \alpha^{\mathcal{M}'}$. We show that $\psi \in H(w)$. There exists a word $\delta = \omega_1\ldots\omega_k \in \mathcal{L}(\alpha)$ such that $(w_0,w) \in \delta^{\mathcal{M}'}$. Let $1 \le i_1 < \ldots < i_h \le k$ be all the indices such that, for $1 \le j \le h$, ω_{i_j} is of the form $(\psi_{i_j}?)$. For $1 \le i \le k$ such that $i \notin \{i_1,\ldots,i_h\}$, let $\omega_i = \varsigma_i$. There exist $w_1,\ldots,w_k \in W$ such that $w_k = w$ and, for $1 \le i \le k$, if ω_i is ς_i then $(w_{i-1},w_i) \in \varsigma_i^{\mathcal{M}'}$, else ($i \in \{i_1,\ldots,i_h\}$ and $\omega_i = (\psi_i?)$ and) $w_i = w_{i-1}$ and $w_i \in \psi_i^{\mathcal{M}'}$, which, by Lemma 4.8, implies that $\overline{\psi}_i \notin H(w_i)$.

We prove by an inner induction on $0 \le i \le k$ that, there exists a sequence $\triangle_i$ of universal modal operators such that $\triangle_i\psi \in H(w_i)$ and $\omega_{i+1}\ldots\omega_k$ is derivable from the program sequence corresponding to $\triangle_i$ using a left derivation of $\mathcal{G}(\alpha)$. The induction hypothesis clearly holds for the base case $i = 0$. Suppose that $i > 0$ and the hypothesis holds for $(i-1)$ in the place of i. Let S_{i-1} be the program sequence corresponding to $\triangle_{i-1}$. There exists S_i such that $\omega_i S_i$ is derivable from S_{i-1}, and $\omega_{i+1}\ldots\omega_k$ is derivable from S_i, using left derivations of $\mathcal{G}(\alpha)$. Let $\triangle_i$ be the sequence of universal modal operators corresponding to S_i. Since $\triangle_{i-1}\psi \in H(w_{i-1})$ and $\mathcal{M}$ is saturated, we have that $[\omega_i]\triangle_i\psi \in H(w_{i-1})$. If $w_i = w_{i-1}$ and $\omega_i = (\psi_i?)$, then $[\psi_i?]\triangle_i\psi \in H(w_i)$, and hence $\triangle_i\psi \in H(w_i)$, since $\overline{\psi}_i \notin H(w_i)$. Consider the other case when $\omega_i = \varsigma_i$. We have that $[\varsigma_i]\triangle_i\psi \in H(w_{i-1})$. Since $(w_{i-1},w_i) \in \varsigma_i^{\mathcal{M}'}$, either $(w_{i-1},w_i) \in R_{\varsigma_i}$ or $(w_i,w_{i-1}) \in R_{\varsigma_i{}^-}$. If $(w_{i-1},w_i) \in R_{\varsigma_i}$, then $\triangle_i\psi \in H(w_i)$ follows from $[\varsigma_i]\triangle_i\psi \in H(w_{i-1})$. If $(w_i,w_{i-1}) \in R_{\varsigma_i{}^-}$, then by the assertion made at the beginning of this proof, $\triangle_i\psi \in H(w_i)$ also follows from $[\varsigma_i]\triangle_i\psi \in H(w_{i-1})$. This completes the inner induction.

As a consequence, we have that $\psi \in H(w_k)$, which means $\psi \in H(w)$. By the assumption of the outer induction, it follows that $\mathcal{M}', w \models \psi$. Therefore $\mathcal{M}', w_0 \models \varphi$, which completes the proof. ◁

4.3 Completeness

Definition 4.10. Let G be an "and-or" graph for (X,Γ) with a consistent marking G' and let v be a node of G'. A *saturation path* of v w.r.t. G' is a finite sequence $v_0 = v$, v_1, …, v_k of nodes of G', with $k \ge 0$, such that, for every $0 \le i < k$, v_i is an "or"-node and (v_i,v_{i+1}) is an edge of G', and v_k is an "and"-node. ◁

Lemma 4.11. *Let G be an "and-or" graph for (X, Γ) with a consistent marking G'. Then each node v of G' has a saturation path w.r.t. G'.*

Proof. We construct a saturation path $v_0, v_1, \ldots$ of v w.r.t. G' as follows. Set $v_0 = v$ and $i = 0$. While v_i is not an "and"-node do:

- If the principal of the static rule expanding v_i is not of the form $\langle\alpha\rangle\varphi$ then let v_{i+1} be any successor of v_i that belongs to G' and set $i := i + 1$.
- If the principal of the static rule expanding v_i is of the form $\langle\alpha\rangle\varphi$ then:
  - let $v_{i+1}, \ldots, v_j$ be the longest sequence of "or"-nodes of G' 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 $\Diamond$-realization in G' for $\langle\alpha\rangle\varphi$ at v_i;
  - set $i := j$.

The loop terminates because each formula not of the form $\langle\alpha\rangle\varphi$ may be reduced at most once. ◁

Lemma 4.12 (Completeness). *Let X and Γ be finite sets of formulas in NCNF of the base language, and let G be an "and-or" graph for (X, Γ). Suppose that G has a consistent marking G'. Then X is satisfiable w.r.t. the set Γ of global assumptions.*

Proof. We construct a model graph $\mathcal{M} = \langle W, (R_\varsigma)_{\varsigma\in\Pi_1}, H\rangle$ as follows:

1. Let v_0 be the root of G' and $v_0, \ldots, v_k$ be a saturation path of v_0 w.r.t. G'. Set $R_\varsigma = \emptyset$ for all $\varsigma \in \Pi_1$ and set $W = \{\tau\}$, where τ is a new node. Set $H(\tau) := \mathcal{L}(v_k) \cup \mathit{rfs}(v_k)$. Mark τ as *unresolved* and set $f(\tau) = v_k$. (Each node of $\mathcal{M}$ will be marked either as unresolved or as resolved, and f will map each node of $\mathcal{M}$ to an "and"-node of G'.)
2. While W contains unresolved nodes, take one unresolved node w_0 and do:
   (a) For every $\langle\varsigma\rangle\langle\alpha_1\rangle\ldots\langle\alpha_h\rangle\varphi \in H(w_0)$, where φ is not of the form $\langle\beta\rangle\psi$, do:
      i. Let $\varphi_0 = \langle\varsigma\rangle\langle\alpha_1\rangle\ldots\langle\alpha_h\rangle\varphi$, $\varphi_i = \langle\alpha_i\rangle\ldots\langle\alpha_h\rangle\varphi$ for $1 \le i \le h$, and $\varphi_{h+1} = \varphi$. Let $u_0 = f(w_0)$. (As a maintained property of f, φ_0 belongs to the label of u_0.) Let the sequence $(u_0, \varphi_0), (u_1, \varphi_1)$ be a $\Diamond$-realization in G' for φ_0 at u_0. Let $i_1 = 1$. For $1 \le l \le h$, let the sequence $(u_{i_l}, \varphi_l), \ldots, (u_{i_{l+1}}, \varphi_{l+1})$ be a $\Diamond$-realization in G' for φ_l at u_{i_l}. Let $u_{i_{h+1}}, \ldots, u_m$ be a saturation path of $u_{i_{h+1}}$ w.r.t. G'.
      ii. Let $j_0 = 0 < j_1 < \ldots < j_{n-1} < j_n = m$ be all the indices such that, for $0 \le j \le m$, u_j is an "and"-node of G iff $j \in \{j_0, \ldots, j_n\}$. For $0 \le s \le n-1$, let $\langle\varsigma_s\rangle\psi_s$ be the label of the edge (u_{j_s}, u_{j_s+1}) of G'. (We have that $\varsigma_0 = \varsigma$.)
      iii. For $1 \le s \le n$ do:
         A. Let $Z_s = \mathcal{L}(u_{j_s}) \cup \mathit{rfs}(u_{j_s})$.
         B. 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 set $f(w_s) = u_{j_s}$.
         C. Add the pair (w_{s-1}, w_s) to $R_{\varsigma_{s-1}}$.
   (b) 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 the closure $\mathit{cls}(X \cup \Gamma)$, 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', which is a trace of φ_0 at u_0 that ends with φ at u_m, to a chain $w_0, \ldots, w_n$ of nodes of

$\mathcal{M}$ by sticking together nodes in every maximal saturation path and using both the sets $\mathcal{L}(u_i)$ and $rfs(u_i)$. Hence, $\mathcal{M}$ is saturated and satisfies the local and global consistency properties. That is, $\mathcal{M}$ is a consistent and saturated model graph.

Consider Step 1 of the construction. As the label of v_0 is $X \cup \Gamma$, we have that $X \subseteq H(\tau)$ and $\Gamma \subseteq H(\tau)$. Consider Step 2(a)iii 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, the label of $u_{j_{s-1}+1}$ contains Γ, and hence the set $\mathcal{L}(u_{j_s}) \cup rfs(u_{j_s})$ also contains Γ. Hence $\Gamma \subseteq H(w_s)$ for every $w_s \in W$. By Lemma 4.9, the Kripke model corresponding to $\mathcal{M}$ validates Γ and satisfies X at τ. Hence, X is satisfiable w.r.t. Γ. ◁

5 An ExpTime Decision Procedure for CPDL

Let X and Γ be finite sets of formulas in NCNF of the base language. In this section, we present a simple ExpTime algorithm for checking satisfiability of X w.r.t. the set Γ of global assumptions. Optimizations for the algorithm are discussed in the next section.

Definition 5.1. Let G be an "and-or" graph for (X, Γ), and G' be a marking of G. The *graph G_t of traces of G' in G* is defined as follows:

- nodes of G_t are pairs (v, φ), where v is a node of G and φ is a formula of the label of v,
- a pair $((v, \varphi), (w, \psi))$ is an edge of G_t if v is a node of G', φ is of the form $\langle\alpha\rangle\xi$, and the sequence (v, φ), (w, ψ) is a trace of φ in G'.

A node (v, φ) of G_t is an *end node* if φ is not of the form $\langle\alpha\rangle\xi$. A node of G_t is *productive* if there is a path connecting it to an end node. ◁

Consider now Algorithm 1, provided in Figure 3, for checking satisfiability of X w.r.t. Γ. The algorithm starts by constructing an "and-or" graph G with root v_0 for (X, Γ). After that it collects the nodes of G whose labels are unsatisfiable w.r.t. Γ. Such nodes are said to be *unsat* and kept in the set $UnsatNodes$. Initially, if G contains a node with label $\{\bot\}$ then the node is *unsat*. When a node or a number of nodes become *unsat*, the algorithm propagates the status *unsat* backwards through the "and-or" graph using the procedure *updateUnsatNodes* (see Figure 3). This procedure has the property that after its execution if the root v_0 of G does not belong to $UnsatNodes$ then the maximal subgraph of G without nodes from $UnsatNodes$, denoted by G', is a marking of G. After each execution of *updateUnsatNodes*, the algorithm finds the nodes of G' that make the marking not satisfying the global consistency property. Such a task is done by creating the graph G_t of traces of G' in G and finding nodes v of G' such that the label of v contains a formula of the form $\langle\alpha\rangle\varphi$ but $(v, \langle\alpha\rangle\varphi)$ is not a productive node of G_t. If the set V of such nodes is empty then G' is a consistent marking (provided that $v_0 \notin UnsatNodes$) and the algorithm stops with a positive answer. Otherwise, V is used to update $UnsatNodes$ by calling $updateUnsatNodes(G, UnsatNodes, V)$. After that call, if $v_0 \in UnsatNodes$ then the algorithm stops with a negative answer, else the algorithm repeats the loop of collecting *unsat* nodes. Note that, we can construct G_t only the first time and update it appropriately each time when $UnsatNodes$ is changed.

Define the *length* of a formula φ to be the number of symbols occurring in φ, and the *size* of a finite set of formulas to be the length of the conjunction of its formulas.

Algorithm 1
Input: finite sets X and Γ of formulas in NCNF of the base language.
Output: *true* if X is satisfiable w.r.t. Γ, and *false* otherwise.

1. construct an "and-or" graph G with root v_0 for (X, Γ);
2. $UnsatNodes := \emptyset$;
3. if G contains a node v with label $\{\bot\}$ then
   $updateUnsatNodes(G, UnsatNodes, \{v\})$;
4. if $v_0 \in UnsatNodes$ then return *false*;
5. let G' be the maximal subgraph of G without nodes from $UnsatNodes$;
   (we have that G' is a marking of G)
6. construct the graph G_t of traces of G' in G;
7. while $v_0 \notin UnsatNodes$ do:
   (a) let V be the set of all nodes v of G' such that the label of v contains a formula of the form $\langle\alpha\rangle\varphi$ but $(v, \langle\alpha\rangle\varphi)$ is not a productive node of G_t;
   (b) if $V = \emptyset$ then return *true*;
   (c) $updateUnsatNodes(G, UnsatNodes, V)$;
   (d) if $v_0 \in UnsatNodes$ then return *false*;
   (e) let G' be the maximal subgraph of G without nodes from $UnsatNodes$;
   (we have that G' is a marking of G)
   (f) update G_t to the graph of traces of G' in G;

Procedure $updateUnsatNodes(G, UnsatNodes, V)$
Input: an "and-or" graph G and sets $UnsatNodes$, V of nodes of G,
where V contains new *unsat* nodes.
Output: a new set $UnsatNodes$.

1. $UnsatNodes := UnsatNodes \cup V$;
2. while V is not empty do:
   (a) take out a node v from V;
   (b) for every father node u of v, if $u \notin UnsatNodes$ and either u is an "and"-node or u is an "or"-node and all the successor nodes of u belong to $UnsatNodes$ then add u to both $UnsatNodes$ and V;

Fig. 3. Algorithm for checking satisfiability of X w.r.t. Γ.

Lemma 5.2. *Let X and Γ be finite sets of formulas in NCNF of the base language, n be the size of $X \cup \Gamma$, and G be an "and-or" graph for (X, Γ). Then G has $2^{O(n^2)}$ nodes and the sets $\mathcal{L}(v)$ and $rfs(v)$ of each node v contain at most $O(n^2)$ formulas.*

Proof. Just note that the sets $\mathcal{L}(v)$ and $rfs(v)$ of each node v of G are subsets of the closure $cls(X \cup \Gamma)$, which contains at most $O(n^2)$ formulas (cf. [13, Lemma 6.3]), and the graph is constructed using global caching. ◁

Lemma 5.3. *Algorithm 1 runs in exponential time in the size of $X \cup \Gamma$.*

Proof. By Lemma 5.2, the graph G can be constructed in $2^{O(n^2)}$ steps and has $2^{O(n^2)}$ nodes, where n is the size of $X \cup \Gamma$. As the sets $\mathcal{L}(v)$ and $rfs(v)$ of each node v of G contain at most $O(n^2)$ formulas, each time when $UnsatNodes$ is extended G_t can be constructed or updated in $2^{O(n^2)}$ steps. Computing the set V can be done in polynomial time in the size of G_t, and hence also in $2^{O(n^2)}$ steps. An execution of $updateUnsatNodes$

is done in polynomial time in the size of G, and hence also in $2^{O(n^2)}$ steps. As the set $UnsatNodes$ is extended at most $2^{O(n^2)}$ times, the total time for executing Algorithm 1 is of rank $2^{O(n^2)}$. ◁

Theorem 5.4. *Let X and Γ denote finite sets of formulas in NCNF of the base language. Algorithm 1 is an* ExpTime *decision procedure for checking whether X is satisfiable w.r.t. the set Γ of global assumptions.*

Proof. It is easy to show that the algorithm has the invariant that a consistent marking of G cannot contain any node of $UnsatNodes$. The algorithm returns *false* only when the root v_0 belongs to $UnsatNodes$, that is, only when G does not have any consistent marking. At Step 7b, G' is a marking of G that satisfies the local consistency property. If at that step $V = \emptyset$ then it satisfies also the global consistency property and is thus a consistent marking of G. That is, the algorithm returns *true* only when G has a consistent marking. Therefore, by Theorem 3.10, Algorithm 1 is a decision procedure for the considered problem. The complexity was established by Lemma 5.3. ◁

6 Some Optimizations

In this section we discuss optimizations for Algorithm 1 given in the previous section. Observe that the algorithm first constructs an "and-or" graph and then checks whether the graph contains a consistent marking. To speed up the performance these two tasks can be done concurrently. For this we update the structures $UnsatNodes$, G', G_t mentioned in the algorithm "on-the-fly" during the construction of G. The main changes are as follows:

- During the construction of the "and-or" graph G, each node of G has status *unexpanded*, *expanded*, *unsat* or *sat*. The initial status of a new node is *unexpanded*. When a node is expanded, we change its status to *expanded*. The status of a node changes to *unsat* (respectively *sat*) when there is an evidence that the label of the node is unsatisfiable (respectively satisfiable) w.r.t. Γ. When a node becomes *unsat*, we insert it into the set $UnsatNodes$.
- When a node of G is expanded or G' is modified, we update G_t appropriately.
- When a new node is created, if its label contains $\bot$ or a clashing pair φ, $\overline{\varphi}$ then we change the status of the node to *unsat*. This is the implicit application of the rule $(\bot_0)$ and a generalized form of the rule $(\bot)$. Thus, we can drop the explicit rules $(\bot_0)$ and $(\bot)$. When a non-empty set V of nodes of G becomes *unsat*, we call $updateUnsatNodes(G, UnsatNodes, V)$ to update the set $UnsatNodes$.
- When $UnsatNodes$ is modified, we update G' appropriately.
- Since G_t is not completed during the construction, when computing the set V of nodes of G' that cause G' not satisfying the global consistency property as in Step 7a of Algorithm 1 we treat a node (v, φ) of G_t also as an *end-node* if v has status *unexpanded* or *sat*.[9] We compute such a set V occasionally, accordingly to some criteria, and when G_t has been completed. The computation is done by propagating "productiveness" backward through the graph G_t. The nodes of the resulting V become *unsat*.

[9] Note that if v has status *unexpanded* (respectively *sat*) then (v, φ) may (respectively must) be a productive node of G_t.

During the construction of the "and-or" graph G, if a subgraph of G has been fully expanded in the sense that none of its nodes has status *unexpanded* or has a descendant node with status *unexpanded* then each node of the subgraph can be determined to be *unsat* or *sat* regardlessly of the rest of G. That is, if a node of the subgraph cannot be determined to be *unsat* by the operations described in the above list then we can set its status to *sat*. This technique was proposed in [16].

Recently, the first author has implemented a tableau prover called TGC (Tableau with Global Caching) [16] for checking consistency of a concept w.r.t. a TBox in the description logic $\mathcal{ALC}$. He has developed and implemented for TGC a special set of optimizations that co-operates very well with global caching and various search strategies on search spaces of the form "and-or" graph. Apart from search strategies and global caching for nodes of the constructed "and-or" graph, TGC also uses other optimizations like normalizing formulas, caching formulas using an efficient catalogue, simplification, semantic branching, propagation of *unsat* in a local scale using *unsat*-cores and subset-checking for parent nodes and brother nodes, as well as cutoffs. The test results of TGC on the sets T98-sat and T98-kb of DL'98 Systems Comparison are comparable with the test results of the best systems DLP-98 and FaCT-98 that took part in that comparison (see [16]). One can say that the mentioned test sets are not representative for practical applications, but the comparison at least shows that various optimization techniques can be applied together with global caching to significantly increase efficiency of tableau decision procedures for modal and description logics.

Most of the optimization techniques of TGC can be applied for our decision procedure for $\mathcal{C}_{\text{CPDL}}$. However, a few things need be further worked out. The first one is how to efficiently compute "*unsat*-core" of a node that becomes *unsat* because it violates the global consistency property.[10] The second one is what normalized form should be used for formulas in CPDL. It is not difficult to give some solutions for these two problems, but their usefulness should be estimated by tests. Despite the fact that the applicability of our rule (*cut*) is quite restricted, the rule is inflexible. It is possible that one can work out a more sophisticated condition for the applicability of the rule (*cut*). On the implementation level, we hope that depth-first search together with propagation of *unsat* for parent/brother nodes and cutoffs significantly reduces the negative side effects of cuts. If this is not the case, one can try to delay cuts in an appropriate way (preserving completeness).

7 Conclusions

We have presented a novel tableau calculus and an optimal (ExpTime) tableau decision procedure based on the calculus for the satisfiability problem of propositional dynamic logic with converse. Our procedure can be implemented together with various useful optimization techniques like the ones of TGC [16].

Our decision procedure for CPDL substantially differs from the procedure given by De Giacomo and Massacci [3] for CPDL:

[10] An *unsat*-core of a node is a subset of the label of the node that causes the node *unsat*. The smaller an *unsat*-core, the better its usefulness (for subset-checking).

- Our decision procedure has optimal complexity ExpTime, while the formal decision procedure given by De Giacomo and Massacci [3] has non-optimal complexity NExpTime. The method described in [3] for transforming their procedure to an ExpTime version is informal, unclear, and has not been proved sound.
- Our decision procedure uses traditional (unlabeled) tableau rules, while the decision procedure of [3] uses labeled (prefixed) tableau rules.
- Our cut rule is of the kind of "guessing the future", which allows global caching in the natural way, while the cut rule used in [3] is of the kind "look behind" for modifying labels of ancestor nodes and it is not clear how to combine that kind of cut with global caching. Furthermore, the applicability of our cut rule is much more restricted than that of the cut rule of [3], which would make our decision procedure more efficient when implemented.

Our decision procedure for CPDL exploits the ideas of global caching and fulfilling eventualities of Pratt's work [20] on PDL, but we have formulated tableaux directly as "and-or" graphs, which makes our calculus and decision procedure for CPDL simpler and easier to understand and implement. Also note that our extension for dealing with converse is indeed nontrivial. In [19], Pratt wrote "We do not have a practical approach to this difficulty with converse, and our practical procedure therefore does not deal with converse."

References

1. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
2. D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 1581–1634. Elsevier, 2001.
3. 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.
4. P. Doherty, B. Dunin-Kęplicz, and A. Szałas. Dynamics of approximate information fusion. In M. Kryszkiewicz, J. Peters, H. Rybinski, and A. Skowron, editors, *Proc. RSEISP 2007*, number 4585 in LNAI, pages 668–677. Springer-Verlag, 2007.
5. F. Donini and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124:87–138, 2000.
6. M.J. Fischer and R.E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.
7. 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.
8. R. Goré. Tableau methods for modal and temporal logics. In D'Agostino et al, editor, *Handbook of Tableau Methods*, pages 297–396. Kluwer, 1999.
9. 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.
10. 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.
11. 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, LNAI 5056*, pages 268–287. Springer-Verlag, 2008.
12. R. Goré and L.A. Nguyen. Sound global caching for abstract modal tableaux. In H.-D. Burkhard et al, editor, *Proceedings of CS&P'2008*, pages 157–167, 2008.

13. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
14. I. Horrocks and P.F. Patel-Schneider. Optimizing description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
15. L.A. Nguyen. Analytic tableau systems and interpolation for the modal logics KB, KDB, K5, KD5. *Studia Logica*, 69(1):41–57, 2001.
16. L.A. Nguyen. An efficient tableau prover using global caching for the description logic ALC. *Fundamenta Informaticae*, 93(1-3):273–288, 2009.
17. L.A. Nguyen and A. Szałas. An optimal tableau decision procedure for converse-PDL. Accepted for KSE'09, 2009.
18. L.A. Nguyen and A. Szałas. A tableau calculus for regular grammar logics with converse. In R.A. Schmidt, editor, *Proceedings of CADE-22, LNAI 5663*, pages 421–436. Springer-Verlag, 2009.
19. V.R. Pratt. A practical decision method for propositional dynamic logic: Preliminary report. In *Proceedings of STOC'1978*, pages 326–337. ACM, 1978.
20. V.R. Pratt. A near-optimal method for reasoning about action. *J. Comput. Syst. Sci.*, 20(2):231–254, 1980.
21. W. Rautenberg. Modal tableau calculi and interpolation. *JPL*, 12:403–423, 1983.
22. K. Schild. A correspondence theory for terminological logics: Preliminary report. In J. Mylopoulos and R. Reiter, editors, *Proceedings of IJCAI'1991*, pages 466–471. Morgan Kaufmann, 1991.
23. R.A. Schmidt. `http://www.cs.man.ac.uk/~schmidt/pdl-tableau/`.
24. M.Y. Vardi. The taming of converse: Reasoning about two-way computations. In R. Parikh, editor, *Logic of Programs, LNCS 193*, pages 413–423. Springer, 1985.