# Tableaux and Algorithms for Propositional Dynamic Logic with Converse

Giuseppe De Giacomo[1] and Fabio Massacci[1,2]

[1] Dip. di Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy,
{degiacomo,massacci}@dis.uniroma1.it
[2] Computer Laboratory, University of Cambridge,
Pembroke Street,Cambridge CB2 3QG, UK,
Fabio.Massacci@cl.cam.ac.uk

**Abstract.** This paper presents a prefixed tableaux calculus for Propositional Dynamic Logic with Converse based on a combination of different techniques such as prefixed tableaux for modal logics and model checkers for mu-calculus. We prove the correctness and completeness of the calculus and illustrate its features. We also discuss the transformation of the tableaux method (naively NEXPTIME) into an EXPTIME algorithm.

## 1 Introduction

Propositional Dynamic Logics (PDL) were used in [4] to describe the properties of states reached by programs during their execution, and to model the evolution of the computation process [10, 16]. Over the years, propositional dynamic logics have been proved to be a valuable theoretical tool in Computer Science, Logic, Computational Linguistics, and Artificial Intelligence (e.g. [1, 6, 9, 10, 16, 18, 15]). Many inference procedures, decidability and complexity results, rely on research done within PDLs.

In this paper we present a tableaux calculus for the propositional dynamic logic *Converse-PDL* ($CPDL$) [4], obtained from the basic logic $PDL$ by adding the converse of a program whose running is obtained by "running the original program backwards". Typically, the converse is used for preconditions e.g. $[\pi^-]\varphi$ can be interpreted as "before running program $\pi$, property $\varphi$ must hold".

There are several applications of PDLs where the ability of denoting converse programs is essential. For instance using PDLs as a core reasoning paradigm of Knowledge Representation Systems. Several recent papers (starting from [15]) point out a strong correspondence between PDLs and a family of class-based knowledge representation formalisms, Description Logics [20]. These logics represent the world in terms of objects grouped into classes, relations between classes and a number of constructs for properties of classes and relations. The correspondence is based on a mapping between the models of a description logic knowledge base, and the models of a particular formula of a propositional dynamic logic, so that classes correspond to propositional letters, relations correspond to atomic programs, instances of classes correspond to states, and instances of relations

correspond to state transitions. Thus, inference procedures for $CPDL$ can be exploited as the reasoning core of very expressive description logics (and PDLs) by using polynomial reductions from the inference problem of such logics to the inference problem of $CPDL$ [2, 3]. This was one of the main motivation that has led us to look into inference procedures for $CPDL$.

$CPDL$ shares many characteristics with the basic $PDL$, and many results for $PDL$ extend to $CPDL$ without difficulties. For instance the proofs of finite model property for $PDL$ in [4] are easily extended to $CPDL$, as well as the proof of EXPTIME-completeness in [13]. However, efficient – in practical cases – inference procedures have been successfully developed for $PDL$, but their extension to $CPDL$ has proved to be a difficult task and unsuccessful till now (to the best of our knowledge).

To be more precise, inference procedures based on models enumeration [4, 13] or on automata on infinite trees [19] have been extended to accommodate converse of programs. Yet, these procedures are better suited for proving theoretical results than for being used in applications. Tableau procedures for $PDL$ [12, 14], which are much more efficient in practice, have never been extended.

The key point is that a tableau procedure for $PDL$ can be organised so that, once the successors a state have been generated, no more reasoning involving this state is necessary. In the case of a $CPDL$ direct extensions of PDL procedures may require reasoning with the whole piece of model built so far. In [12] Pratt says "We do not have a practical approach to this difficulty with converse, and our "practical" procedure therefore does not deal with converse".

Our solution is to use labelled deduction [7] to develop modal prefixed tableaux [5, 8, 11] for $CPDL$. In particular we use Single Step Tableaux [11, 8] since they make it possible to reason locally both in "forward" and "backward" directions to accommodate the converse. The presence of the iteration operator imposes further constraints which lead to the notion of ignorable branches: branches which are modally consistent but where some iterated eventualities are never fulfilled.

The difficulties due to the combination of iteration and converse are solved by singling out few additional formulae that "anticipate" the properties that a state may require from its predecessors at a later stage of the computation.

The next section introduces preliminaries and proof theory in presented in Sect. 3. Examples are shown in Sect. 4, soundness and completeness are given in Sect. 5 and the transformation of NEXPTIME tableaux into EXPTIME algorithm is sketched in Sect. 6. Finally Sect. 7 concludes the paper.

## 2 Preliminaries

We briefly present the basic notions on $CPDL$ (see [10, 16] for surveys).

Let $\mathcal{A}$ be a set of atomic programs and $\mathcal{P}$ a set of propositional letters, the language of $CPDL$ is constructed as follows, where $P \in \mathcal{P}$ and $a \in \mathcal{A}$:

$$\varphi, \psi ::= P \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle\rho\rangle\varphi$$
$$\rho, \chi ::= a \mid \rho; \chi \mid \rho \cup \chi \mid \rho^* \mid \rho^- \mid \varphi?$$

Other connectives, such as $\varphi \vee \psi$ and $[\rho]\varphi$ can be seen as abbreviations – e.g. $[\rho]\varphi \equiv \neg\langle\rho\rangle\neg\varphi$. Without loss of generality, we restrict the application of the converse operator to atomic programs by using equivalences such as $(\rho;\chi)^- \equiv (\chi^-;\rho^-)$ or $(\rho^*)^- \equiv (\rho^-)^*$ etc. We use the metavariable $A$ to denote either an direct or converse atomic program, assuming that $(a^-)^- \doteq a$. In the sequel $P, Q$ are propositional variables and $\varphi, \psi$ formulae whereas $a, b, c$ atomic programs and $\rho, \chi$ programs. $\Phi$ or $\Psi$ are the formulae to be proved valid or satisfiable.

$CPDL$ semantics is based on transition systems (Kripke structures) [10]: a model is a pair $\langle S, \mathcal{I}\rangle$ where $S$ is a non empty set of states and $\mathcal{I}$ an interpretation such that for every atomic program $a \in \mathcal{A}$ it is $a^{\mathcal{I}} \subseteq S \times S$ and for every propositional letter $P \in \mathcal{P}$ it is $P^{\mathcal{I}} \subseteq S$.

The interpretation $\mathcal{I}$ is extended to $CPDL$ formulae and programs as follows:

$$
\begin{array}{ll}
(\psi \wedge \varphi)^{\mathcal{I}} & = \varphi^{\mathcal{I}} \cap \psi^{\mathcal{I}} \\
(\neg\varphi)^{\mathcal{I}} & = S - \varphi^{\mathcal{I}} \\
(\langle\rho\rangle\varphi)^{\mathcal{I}} & = \{\, s \mid \exists s' \in S \text{ s.t. } \langle s, s'\rangle \in \rho^{\mathcal{I}} \text{ and } s' \in \varphi^{\mathcal{I}} \,\} \\
(\rho;\chi)^{\mathcal{I}} & = \{\, \langle s, s'\rangle \mid \exists s''\langle s, s''\rangle \in \rho^{\mathcal{I}} \text{ and } \langle s'', s'\rangle \in \chi^{\mathcal{I}} \,\} \\
(\rho \cup \chi)^{\mathcal{I}} & = \rho^{\mathcal{I}} \cup \chi^{\mathcal{I}} \\
(\rho^*)^{\mathcal{I}} & = \text{reflexive transitive closure of } \rho^{\mathcal{I}} \\
(\rho^-)^{\mathcal{I}} & = \{\, \langle s', s\rangle \mid \langle s, s'\rangle \in \rho^{\mathcal{I}} \,\} \\
(\varphi?)^{\mathcal{I}} & = \{\, \langle s, s\rangle \mid s \in \varphi^{\mathcal{I}} \,\}
\end{array}
$$

In the sequel we write $s \models \varphi$ for $s \in \varphi^{\mathcal{I}}$.

**Definition 1.** A $CPDL$ formula $\Phi$ is *satisfiable* iff there is a model $\langle S, \mathcal{I}\rangle$ where $(\Phi)^{\mathcal{I}}$ is not empty. A formula $\Phi$ is *valid* if for every model $\langle S, \mathcal{I}\rangle$ it is $(\Phi)^{\mathcal{I}} = S$.

The *Fisher-Ladner closure* of a formula $\Phi$ [4, 10] is defined inductively as:

- $\Phi \in CL(\Phi)$;
- if $\varphi \in CL(\Phi)$ then $\neg\varphi \in CL(\Phi)$, provided $\varphi$ does not start with $\neg$;
- if $\neg\varphi$, $\varphi \wedge \psi$ or $\langle\rho\rangle\varphi$ are in $CL(\Phi)$ then $\varphi, \psi \in CL(\Phi)$;
- if $\langle\rho;\chi\rangle\varphi \in CL(\Phi)$ then $\langle\rho\rangle\langle\chi\rangle\varphi \in CL(\Phi)$;
- if $\langle\rho \cup \chi\rangle\varphi \in CL(\Phi)$ then both $\langle\rho\rangle\varphi$ and $\langle\chi\rangle\varphi$ are in $CL(\Phi)$;
- if $\langle\psi?\rangle\varphi \in CL(\Phi)$ then $\psi \in CL(\Phi)$;
- if $\langle\rho^*\rangle\varphi \in CL(\Phi)$ then $\langle\rho\rangle\langle\rho^*\rangle\varphi \in CL(\Phi)$.

The notion of Fisher-Ladner closure is closely related to the notion of set of subformulae in modal logics: to establish the truth value of a formula $\Phi$ in a model it is sufficient to check the value of the formulae in $CL(\Phi)$ for every state of the model [4, 10]. Both number and size of the formulae in $CL(\Phi)$ are linearly bounded by the size of $\Phi$.

## 3 Proof Theory

Prefixed tableaux for $CPDL$ use *prefixed formulae*, i.e. pairs $\langle\sigma : \varphi\rangle$ where $\sigma$ is an alternating sequence of integers and atomic (direct or converse) programs called *prefix* and $\varphi$ is a $CPDL$ formula.

$$\alpha : \frac{\sigma : \varphi \wedge \psi}{\begin{array}{c}\sigma : \varphi \\ \sigma : \psi\end{array}} \qquad \beta : \frac{\sigma : \neg(\psi \wedge \psi)}{\sigma : \neg\varphi \mid \sigma : \neg\psi} \qquad dneg : \frac{\sigma : \neg\neg\varphi}{\sigma : \varphi}$$

**Fig. 1.** Propositional tableaux rules

$$[seq] : \frac{\sigma : \neg\langle\chi;\rho\rangle\varphi}{\sigma : \neg\langle\chi\rangle\langle\rho\rangle\varphi} \qquad \langle seq\rangle : \frac{\sigma : \langle\chi;\rho\rangle\varphi}{\sigma : \langle\chi\rangle\langle\rho\rangle\varphi}$$

$$[test] : \frac{\sigma : \neg\langle\psi?\rangle\varphi}{\sigma : \neg\psi \mid \sigma : \neg\varphi} \qquad \langle test\rangle : \frac{\sigma : \langle\psi?\rangle\varphi}{\begin{array}{c}\sigma : \psi \\ \sigma : \varphi\end{array}}$$

$$[choice] : \frac{\sigma : \neg\langle\chi \cup \rho\rangle\varphi}{\begin{array}{c}\sigma : \neg\langle\chi\rangle\varphi \\ \sigma : \neg\langle\rho\rangle\varphi\end{array}} \qquad \langle choice\rangle : \frac{\sigma : \langle\chi \cup \rho\rangle\varphi}{\sigma : \langle\chi\rangle\varphi \mid \sigma : \langle\rho\rangle\varphi}$$

**Fig. 2.** Rules for sequence, choice, and test

**Definition 2.** The set of prefixes $\Sigma$ is the least set such that $1 \in \Sigma$, and if $\sigma \in \Sigma$, $A \in \mathcal{A} \cup \mathcal{A}^-$ and $n$ is an integer, then $\sigma\langle A\rangle n \in \Sigma$.

Intuitively $\sigma$ "names" the sequence of atomic programs (or path) to reach the state where $\varphi$ holds. For instance the prefix $1\langle a^-\rangle 2\langle b\rangle 3\langle a\rangle 5\langle c^-\rangle 4$ corresponds to the transition $s_1 \stackrel{a}{\leftarrow} s_2 \stackrel{b}{\rightarrow} s_3 \stackrel{a}{\rightarrow} s_5 \stackrel{c}{\leftarrow} s_4$.

We use the standard initial subsequence ordering $\sqsubseteq$ – i.e. impose $\sigma \sqsubset \sigma\langle A\rangle n$ for every $\sigma$, $A$ and $n$, and take the transitive and reflexive closure.

The definition of branch and tableau are similar (but the rules) to prefixed tableaux for modal logics [5, 8, 11]. A *tableau* is a rooted (binary) tree where nodes are labelled with formulae, and a *branch* is path from the root to a leaf. A prefix is *present* in a branch, if there is a prefixed formula with that prefix already in the branch, and it is *new* if it is not already present. In the sequel $\mathcal{B}$ denotes a branch and $\mathcal{T}$ a tableau. Intuitively a branch is a (tentative) model for the initial formula. Propositional rules are also standard (Fig. 1).

The rules for sequence, choice, and test are also simple (see Fig. 2).

Prefixed $CPDL$ formulae starting with an atomic program $a$ or $a^-$ must take into account not only the classical division of possibility-like formulae $\langle a\rangle\varphi$ and necessity-like formulae $[a]\varphi$ but also the presence of the converse operator. Thus we use *both forward and backward rules* for necessity like subformula, as shown in Fig. 3 (where the subscript $F$ stands for forward and $B$ for backward).

The rules for iteration combine prefixed tableaux with the techniques developed by [17] for model checking in modal mu-calculus, based on the intro-

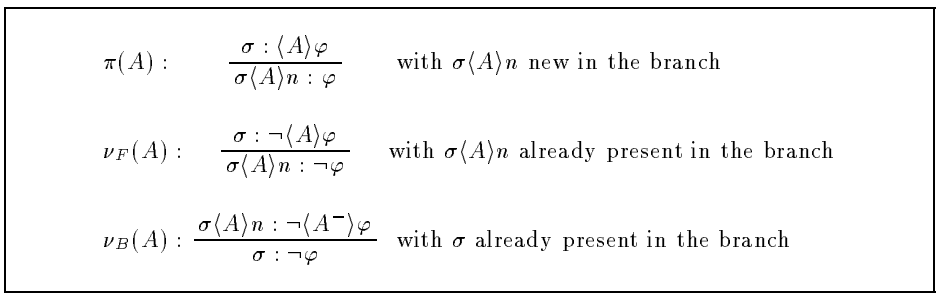


**Fig. 3.** Transitional rules for $CPDL$

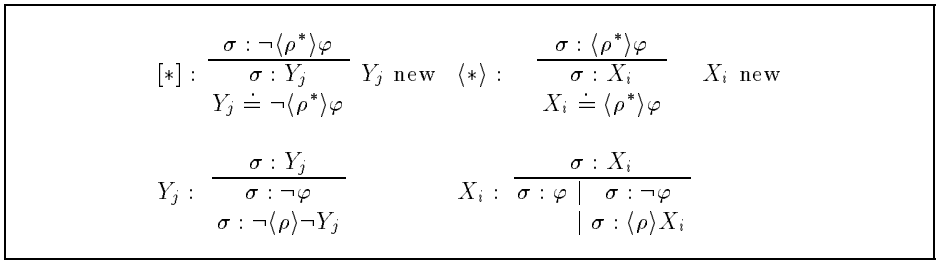


**Fig. 4.** Rules for $*$-iteration operator

duction of constants for fixpoints[3] (Fig. 4). Intuitively the procedure works as follows: when an iterated eventuality $\langle\rho^*\rangle\varphi$ is found, introduce a *new* propositional constant $X_i$, set a side condition $X_i \doteq \langle\rho^*\rangle\varphi$, and use the $X_i$-rule for further reductions. Thus we need two sets of propositional letters distinct from the set $\mathcal{P}$: $\mathcal{X}$ for iterated eventualities and $\mathcal{Y}$ iterated necessities[4]. The use of $\sigma : \neg\varphi$ in the right part of the $X_i$ rule is semantically motivated by the definition of $\langle\rho*\rangle\varphi$ as a *least* fixpoint. Such a definition implies that $\rho$-steps are performed while $\neg\varphi$ is true, stopping as soon as $\varphi$ becomes true. Indeed $\langle\rho^*\rangle\varphi \equiv \langle(\neg\varphi?;\rho)^*\rangle\varphi \equiv \langle\texttt{while}\ \neg\varphi\ \texttt{do}\ \rho\rangle\top$ is valid in $CPDL$ [4].

These constants are introduced to detect the presence of $\rho$ loops which never fulfill $\langle\rho^*\rangle\varphi$, i.e. where $\varphi$ never holds. In this way we can eliminate the $\Rightarrow$ (and its transitive closure) introduced by Pratt's tableaux [12], to relate pseudo models to actual models.

*Remark.* The presence of the converse operator $\cdot^-$ combined with the $\cdot^*$ operator is harder than the simple combination of the two operators: although $\nu_B$ is enough for $CPDL$ without iteration and $X_i/Y_j$ rules are enough for $PDL$, their combination is not enough for full $CPDL$.

[3] A formula $\langle\rho^*\rangle\varphi$ can be expressed in modal mu-calculus as $\mu X.\varphi \vee \langle\rho\rangle X$ while a formula $[\rho^*]\varphi$ can be expressed as $\nu X.\varphi\wedge[\rho]X$, where $\mu X.\Psi(X)$ and $\nu X.\Psi(X)$ denote the least fixpoint and the greatest fixpoint of the open formulae $\Psi(X)$.

[4] Indeed the last one is not really necessary.

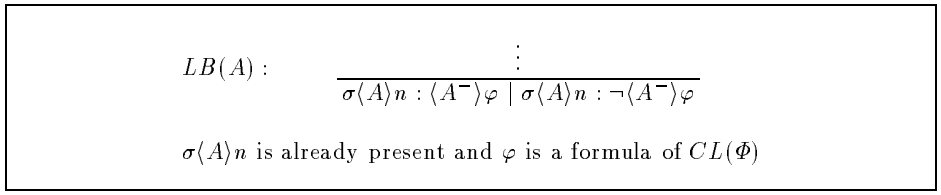


**Fig. 5.** Look behind analytic cut

Intuitively one can use $\cdot^*$ to construct events which take place after *unbounded delays* such as $\langle a^*\rangle P$. The operator $\cdot^-$ can be used for *late discoveries* which impose a property on the current state *after* the execution of a program: such as $\langle a\rangle[a^-]\neg P$. The combination of $\cdot^-$ and $\cdot^*$ can create "bombs" which, after an unbounded number of iterations, tell us that the initial state was inconsistent. A simple unsatisfiable formula is $P \wedge \langle a^*\rangle[(a^-)^*]\neg P$.

So, we use a restricted *analytic cut* LB ("look behind") which is presented in Fig. 5, where $\Phi$ is the formula to be proved valid or satisfiable.

Since the cut is analytic and its application strongly restricted, its introduction does not destroy the decidability of the calculus (although its naive and not necessary application may lead to an explosion of the search space).

Once we have set the rules, we focus on three kinds of branches: *contradictory* (we found a states where $P$ and $\neg P$ are supposed to hold), *ignorable* (we didn't find contradictions but we couldn't fulfill some iterated eventualities) and *open*.

In the sequel, if $\mathcal{B}$ is the branch of a tableau we indicate with $\mathcal{B}/\sigma$ the set of prefixed formulae in $\mathcal{B}$ labelled with the prefix $\sigma$, i.e. :

$$\mathcal{B}/\sigma = \{\varphi \mid \langle \sigma : \varphi\rangle \in \mathcal{B}\}.$$

**Definition 3.** A prefix $\sigma$ is *reduced* if $\pi$-rules are the only rules which have not been applied to formulae of $\mathcal{B}/\sigma$. It is *fully reduced* if all rules have been applied.

**Definition 4.** A prefix $\sigma'$ is a *copy* of a prefix $\sigma$ if (i) $\mathcal{B}/\sigma = \mathcal{B}/\sigma'$, and (ii) both have the form $\sigma_0\langle A\rangle n$ and $\sigma_0'\langle A\rangle n'$ for the same atomic program (direct or converse) $A$. In case two $X_i, X_j$ are present in both prefix we assume them equal if they stand for the same iterated eventuality i.e. $X_i \doteq \langle\rho^*\rangle\varphi$ and $X_j \doteq \langle\rho^*\rangle\varphi$.

This definition of a copy is more restrictive than the corresponding definition one needs for simple $PDL$ [14]. Intuitively a copy is "a different name for the same state" since they (i) have the same properties (dynamic formulae) *and* (ii) can be reached by the same program. This requirement is not necessary for $PDL$ (one only looks forward) whereas in $CPDL$ the past does matter.

**Definition 5.** A tableau branch $\mathcal{B}$ is *$\pi$-completed* if (i) all prefixes are reduced, and (ii) for every $\sigma'$ which is not fully reduced there is a (shorter) copy $\sigma$ which is fully reduced.

The intuition behind $\pi$ completeness is that we use $\pi$-rule to create a new state only if we have not seen it before.

**Definition 6.** A tableau branch $\mathcal{B}$ is *contradictory* if it contains both $\sigma : P$ and $\sigma : \neg P$, for some propositional variable $P$ and some prefix $\sigma$.

**Definition 7.** A branch $\mathcal{B}$ is *ignorable* if and only if

1. it is $\pi$-completed,
2. it contains a prefixed formula $\langle \sigma : X_i \rangle$ where $X_i \doteq \langle \rho^* \rangle \varphi$,
3. for every prefix $\sigma'$ such that $\langle \sigma' : X_i \rangle$ is in $\mathcal{B}$, then $\langle \sigma' : \neg\varphi \rangle$ is also in $\mathcal{B}$.

This definition can be better explained by the following property:

**Proposition 8.** *If a branch $\mathcal{B}$ is ignorable due to some $X_i \doteq \langle \rho^* \rangle \varphi$, then there is a prefix $\sigma_\omega$ in $\mathcal{B}$ such that $\sigma_\omega : X_i$ is in $\mathcal{B}$, $\sigma_\omega : \neg\varphi$ is in $\mathcal{B}$ and $\sigma_\omega$ is a copy of a shorter prefix $\sigma_0$.*

Intuitively one may describe this property has follows: we found an eventuality $\langle \rho^* \rangle \varphi$ in the branch; we tried to fulfill it with $X_i$-rules; however the left hand branches (those with $\sigma : \varphi$) were always discarded; finally we met a prefix $\sigma_\omega$ with the same formulae of a previously seen prefix; so we concluded that we could never fulfill the eventuality in this branch and gave up.

This is clearly the critical stage of the proof procedure (discarding bad branches) and in $CPDL$ we must be sure that two prefixes are identical also for what regard the past. This is the only place where we need to use cut:

*Criterion.* The rule $LB(A)$ is applicable iff the prefixed formula $\sigma\langle A \rangle n : X_i$ occurs already in the branch for some $X_i \doteq \langle \rho^* \rangle \varphi$, $\sigma\langle A \rangle n : \neg\varphi$ occurs, and $\sigma\langle A \rangle n$ is a copy of some other prefix.

So before discarding a branch with Defn. 7 we must be sure that cut has been already applied to $\sigma_\omega$ and $\sigma_0$ of Prop. 8. Thus also their past is identical (at least wrt the Fisher-Ladner closure).

**Definition 9.** A branch is *open* if it is $\pi$-completed and neither contradictory nor ignorable.

**Definition 10.** A tableau is *closed* if all branches are either contradictory or ignorable. A tableaux is *open* if at least one branch is open.

**Definition 11.** A *validity tableaux proof* for the formula $\Phi$ in the logic $CPDL$, i.e. $\vdash_{CPDL} \Phi$ is the closed tableau starting with $\langle 1 : \neg\Phi \rangle$.

In a dual way one can define a satisfiability proof.

*Remark.* For satisfiable formulae a model can be easily extracted from the open branch of the tableau, with the same procedure of the completeness proof.

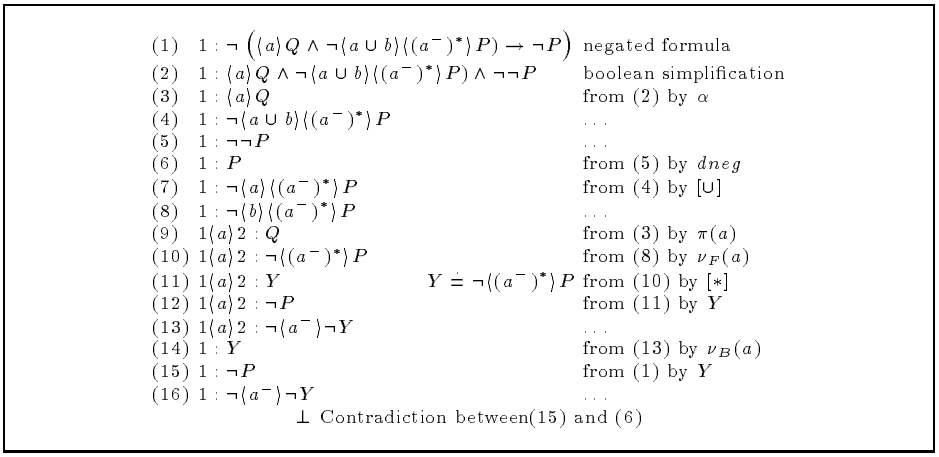


**Fig. 6.** Tableaux proof of $\langle a\rangle Q \wedge \neg\langle a \cup b\rangle\langle(a^-)^*\rangle P) \rightarrow \neg P$

## 4 Examples and Intuitions

A simple example of a tableaux proof for $CPDL$ of the valid formula $\langle a\rangle Q \wedge \neg\langle a \cup b\rangle\langle(a^-)^*\rangle P) \rightarrow \neg P$ is shown in Fig. 6 (numbers are for references).

The intuitions behind "copied prefixes" and ignorable branches can be explained with model theoretic concepts, by comparing tableau rules expansions to a visit of a (counter) model and prefixes to booking devices (names for states).

Whenever we find two prefixes $\sigma_0$ and $\sigma_\omega$ which have the same formulae (i.e. the same properties) we may conclude that they are essentially identical (model $\mathcal{M}$ in Fig. 7). Thus, there is no need to expand the formulae of $\sigma_\omega$: we

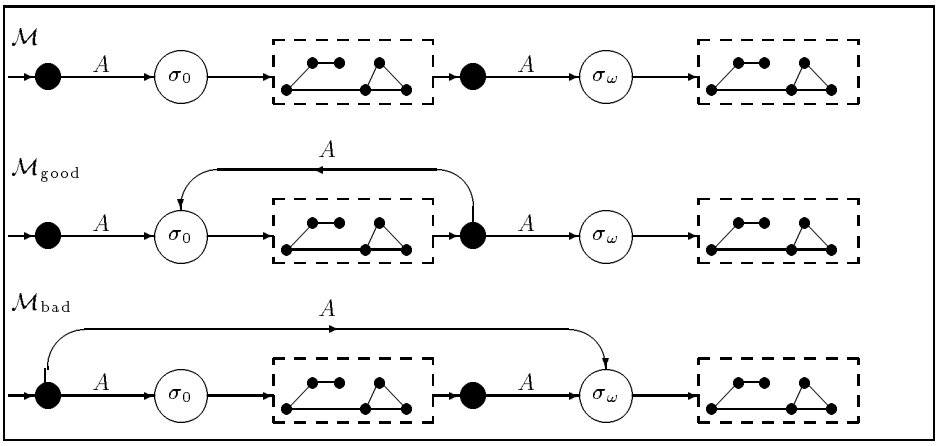


**Fig. 7.** Bad and good models

have already done it for $\sigma_0$ (by Defn. 5) and if we didn't find a contradiction before we will not find it now. We can avoid the visit of the potential infinite path starting from $\sigma_\omega$ by changing the model, according whether the branch is ignorable or not.

If the branch is not ignorable then we introduce a loop back to $\sigma_0$, thus dropping the infinite path starting from $\sigma_\omega$ (model $\mathcal{M}_{\text{good}}$ in Fig. 7).

If the branch is ignorable then there is an eventuality $\langle\rho^*\rangle\phi$ on $\sigma$ that, after a certain number $\langle\rho\rangle$-steps where $\neg\phi$ always holds, arrives to an "identical" state $\sigma_\omega$. So we can change the model to $\mathcal{M}_{\text{bad}}$ (Fig. 7), and conclude that we cannot fulfill the eventuality in any number of $\rho$-steps.

These are the ideas behind the correctness theorem (model $\mathcal{M}_{\text{bad}}$) and the completeness theorem (model $\mathcal{M}_{\text{good}}$). In the tableaux for $PDL$ by Pratt [12, 14] these two cases where called successful and unsuccessful loop.

*Question 12.* Why different $X_i$ are introduced each time the same $\langle\rho^*\rangle\phi$ is met with a different prefix if later on we identify them in the loop checking?

We use the propositional constant $X_i \doteq \langle\rho^*\rangle\phi$ as an automatic bookkeeping system: if we introduce $\sigma : X_i$ at a certain stage and, later on, we find another $\sigma' : X_i$ for a longer $\sigma'$ we already know, without further checks that there is some $\langle\rho\rangle$-steps from $\sigma$ to $\sigma'$. Thus, if $\sigma' : \phi$ is present in the branch, we can immediately conclude that the initial occurrence of the eventuality $\langle\rho^*\rangle\phi$ is fulfilled. If we reused the same variable $X_i$ for a different prefix $\sigma'' : \langle\rho^*\rangle\phi$, then we could not anymore detect whether $\sigma'$ follows from $\sigma$ or from $\sigma''$. Detecting if two prefixes are connected by some $\langle\rho\rangle$-steps has the same complexity of the original problem since $\rho$ may be extremely complicated.

If we find out that $X_i \doteq \langle\rho^*\rangle\phi \doteq X_j$ this means that they are just different names for the same property: if a state $\sigma_0$ fulfils the same formulae of $\sigma_\omega$ plus $X_i$ then it clearly can also fulfill the $X_j$ occurring in $\sigma_\omega$ and thus we can identify the two states (model $\mathcal{M}_{\text{good}}$ of Fig. 7).

For instance try the following (without $LB(A)$, since there is no converse):

$$\Phi_{SAT} \doteq P \wedge [b^*](\langle b\rangle P \wedge \langle a^*\rangle\neg P) \qquad \Phi_{UNSAT} \doteq \Phi_{SAT} \wedge [b^*; a^*]P$$

*Question 13.* Is cut really necessary?

The difficulty is $[(\rho^-)^*]$ which imposes constraints on past computations. For instance check the following formulae without using cut:

$$\begin{aligned}\Psi_{UNSAT} &= P \wedge \langle a^*\rangle[(a^-)^*]\neg P\\ \Psi_{SAT} &= P \wedge \langle a^*\rangle(\neg P \wedge [a^-]\neg P \wedge [a^-; a^-]\neg P \ldots \wedge [(a^-)^n]\neg P)\end{aligned}$$

where we abbreviate $a^-; \ldots; a^-$ for $n$ times with $(a^-)^n$. The second formula is satisfiable, while the first is not. In both cases, if one expands the tableau without using cut, after the first $n$ applications of the $X_i$ rule the resulting tableau will be ignorable. However, after $n+1$ steps, the tableau for $\Psi_{SAT}$ has one non-ignorable branch whereas the one for $\Psi_{UNSAT}$ remains ignorable.

This problem disappear if one uses the uneven version of tableau rules for disjunctive formulae (usually called lemma generation). For instance using $\sigma : \phi \vee \psi$ implies $\sigma : \phi$ or $\sigma : \neg\phi \wedge \psi$. So we propose the following conjecture:

**Conjecture.** *Look behind cut is eliminable for the validity checking of $CPDL$ if lemma generation is used.*

*Remark.* It is easy to prove that cut can be eliminated if the initial formula $\Phi$ contains either only the converse operator or only the iteration operator.

## 5 Soundness and Completeness

The correctness proof of prefixed tableaux [5, 11] follows an established path:

1. devise an mapping between "names" (prefixes) and "things" (states) so that relations between states are preserved;
2. prove a safe extension lemma, i.e. that any tableau rule applied to a satisfiable formula preserve satisfiability with the above mentioned mapping;
3. prove a safe closure lemma, i.e. that the calculus correctly ignores branches which do not correspond to models either because they are contradictory or because do not fulfill some iterated eventuality $\langle\rho^*\rangle\varphi$.

*Remark.* For modal logics safe closure is immediate (a branch must only be non contradictory) whereas it is the hardest part for $(C)PDL$: we have to verify, with a *finite* computation, that an eventuality will *never* be fulfilled.

**Definition 14.** Let $\mathcal{B}$ be a set of prefixed formulae and $\langle S, \mathcal{I}\rangle$ a model, a *mapping* is a function $\imath() : \Sigma \rightarrow S$ such that for all $\sigma$ and $\sigma\langle A\rangle n$ present in $\mathcal{B}$ it is $\langle\imath(\sigma), \imath(\sigma\langle A\rangle n)\rangle \in A^{\mathcal{I}}$ where $A$ is either a direct or converse atomic program.

**Definition 15.** A tableaux branch $\mathcal{B}$ is *satisfiable* (SAT for short) in the model $\langle S, \mathcal{I}\rangle$ if there is a mapping $\imath()$ such that for every $\langle\sigma : \varphi\rangle$ present in $\mathcal{B}$ it is $\imath(\sigma) \models \varphi$. A tableau is SAT if one branch is such for some model $\langle S, \mathcal{I}\rangle$.

**Theorem 16.** *If $\mathcal{T}$ is a SAT tableau, then the tableau $\mathcal{T}'$ obtained by an application of a tableau rule is also SAT.*

*Proof.* By induction on the rules applied as in [5, Chapter 8] or [8, 11].

Now we prove that that ignorable branches can be safely discarded (the key point of the proof). The following preliminary result is useful:

**Lemma 17.** *Let $\mathcal{B}$ be a $\pi$-completed branch and $Path(X_i)$ be the set of prefixes $\sigma$ such that $\langle\sigma : X_i\rangle$ is present in $\mathcal{B}$ then*

1. *$Path(X_i)$ is totally ordered wrt $\sqsubseteq$;*
2. *the prefix $\sigma_0$ where $X_i$ has been firstly introduced is the minimum element;*

3. *if the branch is not ignorable then the prefix $\sigma_\omega$ such that both $\langle\sigma_\omega : X_i\rangle$ and the corresponding $\langle\sigma_\omega : \varphi\rangle$ are present is the maximum element.*

*Proof.* By simple induction on the number of applied tableaux rules: the reduction of $\langle\sigma : \langle\rho\rangle X_i\rangle$ can only introduce prefixes longer (or equal) to $\sigma$. □

**Theorem 18.** *If $\mathcal{T}$ is a SAT tableau, then one SAT branch is not ignorable.*

*Proof.* Suppose the contrary: $\mathcal{T}$ is SAT with all SAT branches ignorable (clearly SAT branches cannot be contradictory). It is worth noting that each branch can be ignorable due to a different unfulfilled $X_i \doteq \langle\rho_i^*\rangle\varphi_i$ (or even more than one). Then let $\mathcal{B}$ be an ignorable branch for $X_i$. It is easy to prove the following

**Proposition 19.** *For every model $\langle S, \mathcal{I}\rangle$ and for every mapping $\imath()$ such that $\mathcal{B}$ is SAT for it, if $\sigma$ is in $Path(X_i)$ then $\imath(\sigma) \not\models \varphi$.*

*Proof.* By definition of ignorable branch (Defn. 7) if $\langle\sigma : X_i\rangle$ is in $\mathcal{B}$ then $\langle\sigma : \neg\varphi_i\rangle$ is also in $\mathcal{B}$. So if $\mathcal{B}$ is SAT on the model $\langle S, \mathcal{I}\rangle$ with mapping $\imath()$ then, by Defn. 15, it is $\imath(\sigma) \models \neg\varphi_i$. □

Since $\mathcal{B}$ is SAT, there must be a model $\langle S, \mathcal{I}\rangle$ and an mapping $\imath()$ on which $\mathcal{B}$ is SAT with a certain mapping $\imath()$.

So let $\sigma_\omega \equiv \sigma'_\omega\langle A\rangle n_\omega$ be the longest prefix such that $\sigma : X_i$ is present. Since $\mathcal{B}$ is $\pi$-completed there must be a shorter copy $\sigma_0 \equiv \sigma'_0\langle A\rangle n_0$ which satisfies the same formulae and which has been fully expanded (Prop.8). Hence the prefixed formula $\langle\sigma_0 : \langle\rho\rangle X_i\rangle$ also occurs in $\mathcal{B}$ and, since $\mathcal{B}$ is SAT, $\imath(\sigma_0) \models \langle\rho\rangle X_i$. Therefore an integer $N$ and a state $s^N$ in $\langle S, \mathcal{I}\rangle$ exist such that $\langle\imath(\sigma_0), s^N\rangle$ is in $(\rho^N)^{\mathcal{I}}$ and $s^N \models \varphi$.

By Lemma 17 each $\sigma'' : X_i$ can only be introduced by reducing the immediate predecessor $\sigma' : \langle\rho\rangle X_1$. Hence, by a simple induction on the structure of $\rho$, there are $R$ $\rho$-step from $\sigma_0$ to $\sigma_\omega$ for some integer $R \geq 1$.

By Prop. 19 $N$ must be strictly greater than $R$ since $\varphi$ cannot be fulfilled by any remapping $\jmath()$ of the $\sigma$ in $Path(X_i)$ on the states of $\langle S, \mathcal{I}\rangle$. Hence there are $N - R$ $\rho$-steps from $\sigma_\omega$ to fulfill $\varphi$ in the model $\langle S, \mathcal{I}\rangle$ under $\imath()$.

Now we construct a new model by duplicating the original model $\langle S, \mathcal{I}\rangle$as in Fig.7: $S' = \{s_c \mid s \in S\}$ and $P^{\mathcal{J}} = \{s_c \mid s \in P^{\mathcal{I}}\}$ and also for atomic program we have $a^{\mathcal{J}} = \{\langle s_c, s'_c\rangle \mid \langle s, s'\rangle \in a^{\mathcal{I}}\}$. The only difference is the atomic (direct or converse) program $A$ in $\sigma_0$ and $\sigma_\omega$: modify $\mathcal{J}$ so that:

$$A^{\mathcal{J}} = \{\langle s_c, s'_c\rangle \mid \langle s, s'\rangle \in A^{\mathcal{I}}\} \cup \{\langle \imath(\sigma'_0)_c, \imath(\sigma_\omega)_c\rangle\}$$

The key point is to prove that this new $A$-arc can be safely added.

Since $\mathcal{B}$ is $\pi$-completed, all possible instances of $LB(A)$ have been applied and therefore for every $\psi \in CL(\Phi)$ we have that either $\langle\sigma_0 : \neg\langle A\rangle\psi\rangle$ or $\langle\sigma_0 : \langle A\rangle\psi\rangle$ is present on the branch. The prefix $\sigma_\omega$ is a copy of $\sigma_0$ by hypothesis, so $\langle\sigma_\omega : \neg\langle A\rangle\psi\rangle$ is present in the branch iff $\langle\sigma_0 : \neg\langle A\rangle\psi\rangle$ is present. Since the branch is SAT on the original model $\langle S, \mathcal{I}\rangle$, it is $\imath(\sigma_0) \models \neg\langle A\rangle\psi$ iff $\imath(\sigma_\omega) \models \neg\langle A\rangle\psi$ for every $\psi \in CL(\Phi)$.

Consider now the state $\imath(\sigma_\omega)_c$ the only difference with the original state $\imath(\sigma_\omega)$ is the incoming $A$-arc. But, as we have seen above, the two states see exactly the same formulae of $CL(\Phi)$ going back through $A$. By the filtration Lemma [4, 10], these are the only formulae necessary for establishing the truth value of $\Phi$. Hence, by induction, we have that $\imath(\sigma_\omega)_c$ satisfies $\langle\rho^*\rangle\varphi$ in $N-R$ $\rho$-steps in the new model (and indeed also in the old one).

Then we construct a new mapping $\jmath()$ on the duplicated model as follows: map every prefix shorter or unrelated with $\sigma_0$ in the same way as $\imath()$ does and $\jmath(\sigma_0)$ on $\imath(\sigma_\omega)_c$. This make the branch still satisfiable: the formulae are the same for both $\sigma_0$ and $\sigma_\omega$ and the incoming arc does not affect them. By Thm. 16. we can expand the tableau and still preserve SAT.

In the new model the state $\jmath(\sigma_0)$ fulfils the eventuality $\langle\rho^*\rangle\varphi$ in $N-R<N$ $\rho$-steps. We can repeat the process until we reach an $N'\leq R$ but this is impossible due to Prop. 19. Contradiction. □

The *correctness theorem* follows with a standard argument:

**Theorem 20.** *If $\Phi$ has a validity proof then $\Phi$ is valid.*

To prove completeness, we also have an established path:

1. apply a systematic procedure to the tableau;
2. if it does not close, choose an open branch to build a model for the initial formula $\neg\varphi$ i.e. a counter-model for $\varphi$;
3. for this construction identify prefixes present in the branch with states and show that if $\langle\sigma:\varphi\rangle$ occurs in the branch then also $\langle\sigma\rangle\models\varphi$. For $PDL$ the hard part of the proof is to show that iterated eventualities are indeed fulfilled.

Then we can prove a *strong model existence theorem* using open branches.

**Theorem 21.** *If $\mathcal{B}$ is an open branch then it is SAT on a $\langle S,\mathcal{I}\rangle$.*

*Proof.* Construct the model as follows:

$$\begin{aligned} S &\doteq \{\sigma \mid \sigma \text{ is present in } \mathcal{B}\\ a^{\mathcal{I}} &\doteq \{\langle\sigma,\sigma\langle a\rangle n\rangle \mid \sigma \text{ and } \sigma\langle a\rangle n \text{ are present in } \mathcal{B}\}\cup\\ &\quad\ \{\langle\sigma\langle a^-\rangle n,\sigma\rangle \mid \sigma \text{ and } \sigma\langle a^-\rangle n \text{ are present in } \mathcal{B}\}\\ P^{\mathcal{I}} &\doteq \{\sigma \mid \sigma : P\in\mathcal{B}\} \end{aligned}$$

To take loops and repetitions into account, we modify slightly the above definition: if $\sigma''$ is a copy of some shorter prefix $\sigma'$ then we delete $\sigma''$ from $S$, replace $\sigma''$ with $\sigma'$ in all transitions $a^{\mathcal{I}}$, and construct the mapping $\imath()$.

$$\imath(\sigma)=\begin{cases}\sigma' \text{ if } \sigma \text{ is a copy of a shorter } \sigma'\\ \sigma \quad \text{otherwise}\end{cases}$$

Next we need to prove that if $\langle\sigma:\varphi\rangle\in\mathcal{B}$ then $\imath(\sigma)\models\varphi$ by induction on the construction of $\varphi$. We focus on modal connectives and iteration operators.

Suppose that $\sigma$ is not a copy of another prefix and $\langle \sigma : \langle a \rangle \varphi \rangle \in \mathcal{B}$ then, by $\pi$-saturation, $\langle \sigma\langle a\rangle n : \varphi \rangle \in \mathcal{B}$ for some $\sigma\langle a\rangle n$. Hence $\imath(\sigma\langle a\rangle n) \models \varphi$ by inductive hypothesis and $\langle \imath(\sigma), \imath(\sigma\langle a\rangle n)\rangle \in a^{\mathcal{I}}$ by construction. Therefore $\imath(\sigma) \models \langle a \rangle \varphi$. If $\sigma$ is a copy there must be a shorter prefix $\sigma'$ present in $\mathcal{B}$ which has the same formulae and which has been fully reduced. In this case the mapping $\imath()$ will map $\sigma$ on $\sigma'$ and the above reasoning applies. Similarly for $a^-$.

For the necessity operator we show the case for $a^-$. Suppose that $\langle \sigma : \neg\langle a^-\rangle \varphi\rangle$ is in $\mathcal{B}$. By construction the only prefixes $\sigma''$ such that $\langle \sigma'', \sigma\rangle \in a^{\mathcal{I}}$ are:

1. $\sigma\langle a^-\rangle n$ for some $n$;
2. $\sigma'$ where $\sigma'$ is a repeated copy of a longer prefix of the form $\sigma\langle a^-\rangle m$;
3. $\sigma_0$ if $\sigma$ has the form $\sigma_0\langle a\rangle m$

For case (1) we have that for every $\sigma\langle a^-\rangle n$ present in $\mathcal{B}$ it is $\langle \sigma\langle a^-\rangle n : \neg\varphi\rangle \in \mathcal{B}$ by $\pi$-completion wrt $\nu_F(a^-)$. Hence, $\imath(\sigma\langle a^-\rangle n) \models \neg\varphi$ by inductive hypothesis. For case (2) the shorter prefix $\sigma'$ must have the same formulae of the copy $\sigma\langle a^-\rangle m$ and, by $\pi$-completion (again the forward rule), we have that $\sigma\langle a^-\rangle m : \neg\varphi$ is present and therefore (it is a copy) also $\sigma' : \neg\varphi$. By induction hypothesis we have that $\imath(\sigma') \models \neg\varphi$. For case (3) consider $\pi$-completion w.r.t. the rule $\nu_B(a^-)$: the prefixed formula $\langle \sigma_0 : \neg\varphi\rangle$ occurs in $\mathcal{B}$. So $\imath(\sigma_0) \models \neg\varphi$ by inductive hypothesis. Therefore, by definition of $\models$, it is $\imath(\sigma) \models \neg\langle a^-\rangle\varphi$.

For the iteration operator the case of $[\rho^*]$ is simple. For $\langle \rho^*\rangle\varphi$ we have to prove that whenever the corresponding $X_i$ appears then $\langle \rho^*\rangle\varphi$ is satisfied. The proof is by double induction: on the formula size and on the length of the prefixes in $Path(X_i)$. One chooses as a base for the latter induction the top prefix $\sigma_\omega$ such that $\langle \sigma_\omega : \varphi\rangle$ is present. By induction hypothesis it is $\imath(\sigma_\omega) \models \varphi$ and by definition it is $\imath(\sigma_\omega) \models \langle \rho^*\rangle\varphi$. For the induction step consider a pair $\sigma_j \sqsubset \sigma_{j+1}$ such that $\sigma_j$ is the immediate predecessor of $\sigma_{j+1}$ in $Path(X_i)$ and that $\imath(\sigma_{j+1}) \models \langle \rho^*\rangle\varphi$. Note that, since $X_i$ was new on the branch, the only way to introduce it for $\sigma_{j+1}$ is to reduce completely $\sigma_j : \langle \rho\rangle X_i$. By induction on the construction of $\rho$ (by using a techniques from [2]) it is possible to verify that $\langle \sigma_j, \sigma_{j+1}\rangle$ is in $\rho^{\mathcal{I}}$ and therefore the claim follows by definition of $\models$. For instance if $\rho \equiv \chi;\tau$ then by $\pi$-completion $\sigma_i : \langle \chi\rangle\langle \tau\rangle X_i$ is on the branch and therefore, by induction, there must be $\sigma'$ such that $\langle \sigma_j, \sigma'\rangle \in \chi^{\mathcal{I}}$ and $\langle \sigma', \sigma_{j+i}\rangle \in \tau^{\mathcal{I}}$ and the claim follow by semantics of sequence operator. □

A *completeness theorem* follows with standard argument:

**Theorem 22.** *If $\Phi$ is valid then $\Phi$ has a validity proof.*

## 6 From NEXPTIME Tableaux to EXPTIME Algorithms

Our tableau leads to the following "naive" algorithm: select a formula from the branch and reduce it; if the reduction requires branching, then choose one branch and add the other to the stack; repeat until the branch is contradictory, ignorable or open; in the first two cases discard the branch and backtrack. This

algorithm compute each time from scratch without keeping track of discarded branches, i.e. *the naive implementation does not learn from failures.* This makes sense for logics in PSPACE [9] but not for $(C)PDL$. In fact the algorithm works in NEXPTIME, while $(C)PDL$ is EXPTIME complete [4, 13].

A smart algorithm can be developed with the techniques of [14]: use a suitable data structure where all possible subsets of the formulae that may appear in the tableau are listed. As soon as our expansion procedures introduces a new formula with a certain prefix, we collect the formulae with the same prefix and look in our database: if this set is already present then we do not expand it further, otherwise we introduce it in the database, *marked with the last atomic (direct or converse) program used to reach it.* This is the difference with [14]: for $CPDL$ two sets must also be equal wrt the "arriving program" (Defn. 5 and Thm. 18). Last we start a marking algorithm which marks bad prefixes as in [14]. A key difference is that we discard at once all prefixes which contains a $X_i$ which makes the branch ignorable. This is more effective than [14] also for $PDL$ since we do not compute the transitive closure of $\Rightarrow$ but just look for $X_i$ locally.

Marking each set with the "arriving programs" and "using cut" implies that, for each atomic programs $A$, our database could contain all propositionally consistent subsets of $\{\psi, \langle A\rangle\psi, \neg\langle A\rangle\psi \mid \psi \in CL(\Phi)\}$. This gives an upper bound for the database size exponential in $O(|Act(\Phi)| \times |\Phi|)$, where $Act(\Phi)$ are the direct or converse atomic programs in $\Phi$, and hence the desired EXPTIME bound.

As a further optimisation, prefixed formulae which branch the tableau or introduce new states are not expanded if one of their reduct is already present in the *same* branch. For instance if $\sigma : \neg\phi$ is already present then $\sigma : \neg(\phi \wedge \psi)$ is not expanded. Similarly if for $\sigma\langle A\rangle n : \phi$ is present then $\sigma : \langle A\rangle\phi$ is not reduced.

## 7 Discussion and Conclusion

Known decision procedures for $CPDL$ are based either on the enumeration of models [4, 13] or on automata on infinite trees [19]. However, these are often inherently exponential. So that the best procedures for $PDL$ are the tableaux methods in [12, 14]. Yet they have not been extended till now.

One characterising feature of $(C)PDL$ is the presence of *fixpoint* operators (the $*$). In comparison with tableaux for modal logics [5, 8, 9, 11], the tableaux for modal fixpoint logics are conceptually divided in two: (1) build a (pseudo) model expanding the modal part; (2) check this model for the satisfiability of fixpoint formulae. The notion of ignorable branches stems out from the idea of merging the second step into the first one.

Such a merging requires to keeps track, during the expansions phase, of iterated eventualities and of their fulfillment. The necessity of (successful and unsuccessful) loop checking for eventualities has been pointed out in [12, 14] for $PDL$, and is even stronger for the modal mu-calculus [17]. For instance in [12] a model checker is run on the final pseudo-model whereas in [14] a new relation symbol $\Rightarrow$ is introduced and some properties of its transitive closure verified.

We think that the use of constants for iterated eventualities, taken from model checking techniques in [17], improves efficiency and readability of the calculus. In this setting our tableaux calculus is a first step towards effective decision procedures for $CPDL$ and the corresponding description logics.

## References


1. P. Blackburn and E. Spaan. A modal perspective on computational complexity of attribute value grammar. *J. of Logic, Language and Information*, 2:129–169, 1993.
2. G. De Giacomo. *Decidability of class-based knowledge representation formalisms.* PhD thesis, Università di Roma "La Sapienza", 1995.
3. G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of AAAI-94*, pages 205–212, 1994.
4. N. J. Fisher and R. E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and System Sciences*, 18:194–211, 1979.
5. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* Reidel, 1983.
6. N. Friedman and J. Halpern. On the complexity of conditional logics. In *Proc. of KR-94*, 1994.
7. D. M. Gabbay. Labelled deductive systems. Tech. Rep. MPI-I-94-223, Max Plank Inst. für Informatik, Saarbrüken, Germany, 1994.
8. R. Goré. Tableaux method for modal and temporal logics. Tech. Rep. TR-ARP-15-5, Australian National University, 1995.
9. J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
10. D. Kozen and J. Tiuryn. Logics of programs. In *Handbook of Theoretical Computer Science*, pages 790–840. Elsevier, 1990.
11. F. Massacci. Strongly analytic tableaux for normal modal logics. In *Proc. of CADE-94*, LNAI 814, pages 723–737. Springer-Verlag, 1994.
12. V. R. Pratt. A practical decision method for propositional dynamic logic. In *Proc. of STOC-78*, pages 326–337, 1978.
13. V. R. Pratt. Models of program logics. In *Proc. of FOCS-79*, pages 115–122, 1979.
14. V. R. Pratt. A near-optimal method for reasoning about action. *J. of Computer and System Sciences*, 20:231–255, 1980.
15. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, 1991.
16. C. Stirling. Modal and temporal logic. In *Handbook of Logic in Computer Science*, pages 477–563. Clarendon Press, 1992.
17. C. Stirling and D. Walker. Local model checking in modal mu-calculus. *Theoretical Computer Science*, 89:161–177, 1991.
18. J. Van Benthem, J. Van Eijck, and V. Stebletsova. Modal logic, transition systems and processes. *J. of Logic and Computation*, 4(5):811–855, 1994.
19. M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Computer and System Sciences*, 32:183–221, 1986.
20. W. A. Woods and J. G. Schmolze. The KL-ONE family. In *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992.


This article was processed using the LaTeX macro package with LLNCS style