ExpTime Tableaux with Global Caching for Hybrid PDL

Linh Anh Nguyen

Received: 5 October 2017 / Accepted: 3 December 2018 / Published online: 1 January 2019
© Springer Nature B.V. 2019

Abstract
We present the first direct tableau decision procedure with the ExpTime complexity for HPDL (Hybrid Propositional Dynamic Logic). It checks whether a given ABox (a finite set of assertions) in HPDL is satisfiable. Technically, it combines global caching with checking fulfillment of eventualities and dealing with nominals. Our procedure contains enough details for direct implementation and has been implemented for the TGC2 (Tableaux with Global Caching) system. As HPDL can be used as a description logic for representing and reasoning about terminological knowledge, our procedure is useful for practical applications.

Keywords Modal logic · Propositional dynamic logic · Tableau-based decision procedures · Global caching · Complexity-optimal tableaux

1 Introduction
Propositional dynamic logic (PDL) [15,24] is one of the most well-known modal logics. It was designed for reasoning about correctness of programs, but can be modified or extended for other purposes. For example, its extension CPDL$_{reg}$ with converse and regular inclusion axioms can be used as a framework for multi-agent logics [14]. As another example, the description logic $\mathcal{ALC}_{reg}$ is a variant of PDL for representing and reasoning about terminological knowledge [36]. Extensions of $\mathcal{ALC}_{reg}$ were also studied by researchers, including the ones with regular inclusion axioms, inverse roles, qualified number restrictions and/or nominals (see, e.g., [10,12,16]).

Automated reasoning in PDL and its extensions is useful for practical applications. The first tableau decision procedure for PDL was developed by Pratt [34]. It implicitly uses global caching and has the ExpTime (optimal) complexity. Nguyen and Szałas [33] reformulated that procedure by explicitly using global caching and extended it for dealing with checking satisfiability of an ABox (a finite set of assertions) in PDL. Goré and Widmann [21] gave another tableau decision procedure with global caching for PDL, which updates fulfillment of
eventualities (i.e., existential star modalities) on-the-fly. Due to global caching, the procedures given in [21,33] have the \( \text{ExpTime} \) complexity.

Tableaux with global caching were formally formulated by Goré and Nguyen for the description logic \( \mathcal{ALC} \) [19] and extended for other logics (see, e.g., [17,22,28,31]). A tableau with global caching for checking satisfiability of a concept w.r.t. a TBox in \( \mathcal{ALC} \) is a rooted “and-or” graph, where the label of each node is a set of concepts treated as requirements to be realized for the node. Using global caching, each node has a unique label, which means that before creating a new node we check whether there already exists a node with the same label that can be used as a proxy. It is sufficient to deterministically construct one “and-or” graph and update the statuses of the nodes by detecting direct clashes and propagating them backward appropriately. Extending tableaux with global caching for PDL [21,33], the ability to check fulfillment of eventualities over the constructed “and-or” graph is essential. Dealing with ABoxes in PDL [33], an “and-or” graph has two kinds of nodes: complex nodes and simple nodes. The label of a complex node is an ABox (with assertions about different states and accessibility between them), while the label of a simple node is a set of formulas (about one state).

Hybrid logics are extensions of ordinary modal logics with an additional sort of symbols, which are called nominals, for labeling states in Kripke models. A nominal is true at exactly one state in a Kripke model. Apart from the operator \(@\) for referring to a nominal (i.e., to a named state), hybrid logics may allow also quantifiers (\( \downarrow \) and \( \forall \)) to bind nominals. For the history and overviews of research on hybrid logics, we refer the reader to [1,9]. The works [3,4,6,8,37] concern the proof theory in hybrid logics. In particular, internalization, a technique that translates metalinguistic formulas to hybrid-logical formulas, has been proposed and studied by Blackburn in [3] and Seligman in [37]. Braüner [8] showed that internalization is a good technique that enables developing proof systems for many modal logics in a uniform way [3,4,6,8]. In [38] Tzakova developed labeled tableau calculi for basic hybrid logics. Since then, internalized tableaux have been developed for hybrid logics (see, e.g., [5–7,11,23]). Several of them have been designed to have termination without loop-checking [5–7,23]. Other notable works on tableau systems for hybrid logics or their variants include: the works [26,27] on automated reasoning in Hybrid PDL, which will be discussed shortly, and the works [25,29,32] on automated reasoning in description logics with nominals. The techniques of [25,26] (resp. [27,29,32]) are related to internalization (resp. “and-or” graphs). The work [32] uses global caching, the work [29] uses “global state caching” [22], and the work [27] “partial caching”.

HPDL (Hybrid PDL) is the modal logic that extends PDL with nominals. In HPDL, nominals are used as formulas, which allow us to refer to concrete states in a Kripke model. For example, the formula \( \langle \sigma \rangle (a \land \varphi) \), where \( \sigma \) is an atomic program and \( a \) is a nominal, represents the fact that the program \( \sigma \) has a transition from the current state to the state named \( a \), which satisfies the formula \( \varphi \). Different occurrences of the same nominal refer to the same state. Thus, a formula \( \downarrow a.\varphi \) in the traditional notation of hybrid logics [1] is written as \( a \land \varphi \). HPDL is more expressive than PDL and belongs to the same complexity class \( \text{ExpTime}\)-complete as PDL [15,35] (regarding the satisfiability problem).

In [26] Kaminski and Smolka developed a decision procedure for HPDL. The authors used the term “goal-directed” to refer to the property that the search is done analytically (i.e., closely based on the input). Like the traditional tableau method for description logics, the search space used in [26] is an “or”-tree of nodes that are “and”-structures, where the “or”-tree is generated by nondeterministic choices (using backtracking) and the “and”-structures are demo graphs (like model graphs or Hintikka structures without statically reduced assertions). Nodes of a demo graph, called “normal clauses” in [26], are formula sets without statically
reduced formulas. Caching is done only within an “and”-structure. This is similar to the technique used in Donini and Massacci’s tableau algorithm for the description logic $ALC$ [13] and somehow equivalent to the “anywhere blocking” technique used in the traditional tableau method for description logics [2]. Without a surprise, the decision procedure given in [26] for HPDL has the NExpTime (non-optimal) complexity for the worst case. Devising efficient ExpTime decision procedures for HPDL and its extensions is claimed in [26] as an open problem.

In [27] Kritsimallis presented a tableau-based algorithm for checking satisfiability of a given formula in the logic $HPDL_\oplus$, which extends HPDL with satisfaction statements (i.e., formulas of the form $\langle a, \varphi \rangle$). His algorithm uses a kind of “and-or” graphs, detects unfulfilled eventualities on-the-fly, and applies partial caching. The caching technique relies on that nodes of a certain kind are available for reuse until they become out of date due to “loop dependencies”. The algorithm has a 2ExpTime complexity for the worst-case.

In this article, we present the first tableau decision procedure with the ExpTime complexity for checking whether a given ABox (a finite set of assertions) in HPDL is satisfiable. Technically, it combines global caching with checking fulfillment of eventualities and the technique of dealing with nominals from our work on tableaux for the description logic $SHIC$ [29]. Global caching not only guarantees the ExpTime complexity, but is also an important optimization technique for increasing efficiency. Our decision procedure for HPDL also uses other advanced techniques, for example, automaton-modal operators.

In the absence of nominals or the ability to express global assumptions, the problem of checking satisfiability of an ABox is usually more general than the problem of checking satisfiability of a formula. In HPDL, the former problem is reducible to the latter. We consider the former instead of the latter due to the nature of our tableau method (which uses both complex nodes and simple nodes for tableaux). Also note that the problem of checking satisfiability of a formula in $HPDL_\oplus$ [27] is reducible to the problem of checking satisfiability of an ABox in HPDL.$^1$

Our decision procedure for HPDL contains enough details for direct implementation and has been implemented for TGC2 [30], which is a system based on tableaux with global caching for automated reasoning in modal and description logics. This system was designed and implemented with several optimization techniques. Regarding memory management, experiments showed that the amount of memory used by TGC2 is competitive with the ones used by the other reasoners. As far as we know, TGC2 is the first implemented system that can be used for reasoning in HPDL. We refer the reader to [30] for details of the design of this system.

The rest of this article is structured as follows. In Sect. 2, we present the syntax and semantics of HPDL and recall automaton-modal operators [14,24]. We omit the feature of “global assumptions” as they can be expressed in PDL (by “local assumptions”). In Sect. 3, we present our tableau calculus for HPDL, starting with the data structure, the tableau rules and ending with the corresponding tableau decision procedure and its properties. In Sect. 4, we

---

$^1$ Let $\psi$ be a formula in $HPDL_\oplus$. Let $a_1, \ldots, a_h$ be all nominals, $\sigma_1, \ldots, \sigma_k$ all atomic programs and $\langle a_{i_1}, \psi_1, \ldots, a_{i_\ell}, \psi_\ell \rangle$ all satisfaction statements occurring in $\psi$. Let $p_1, \ldots, p_\ell$ be new propositions (i.e., atomic formulas) and $\psi', \psi_1', \ldots, \psi_\ell'$ the formulas obtained from $\psi, \psi_1, \ldots, \psi_\ell$, respectively, by replacing each satisfaction statement $\langle a_{i_j}, \psi_j \rangle$ by $p_j$, for $1 \leq j \leq \ell$. Let $\tau_0$ and $\tau$ be new nominals and $\varphi$ a new atomic program. Define $\mathcal{A}$ to be the ABox consisting of the assertions $\tau : \psi', \varphi(\tau_0, \tau), \varphi(\tau_0, a_i)$ (for $1 \leq i \leq h$), $\tau_0 : ([\varphi](a_{i_j} \land \psi_\ell) \rightarrow ([\varphi \cup \sigma_1 \cup \ldots \cup \sigma_k] p_j) \land \tau_0 : ([\varphi](a_{i_j} \lnot \psi_\ell) \rightarrow ([\varphi \cup \sigma_1 \cup \ldots \cup \sigma_k] \lnot p_j) \land \tau_0 : ([\varphi](a_{i_j} \land \psi_\ell) \rightarrow ([\varphi \cup \sigma_1 \cup \ldots \cup \sigma_k] p_j) \land \tau_0 : ([\varphi](a_{i_j} \lnot \psi_\ell) \rightarrow ([\varphi \cup \sigma_1 \cup \ldots \cup \sigma_k] \lnot p_j)$ (for $1 \leq j \leq \ell$). Clearly, $\mathcal{A}$ does not contain satisfaction statements. It can be shown that $\psi$ is satisfiable iff $\mathcal{A}$ is satisfiable.
2 Preliminaries

2.1 Hybrid Propositional Dynamic Logic

We use $\Sigma$ to denote the set of atomic programs, $\mathcal{PROP}$ to denote the set of propositions (i.e., atomic formulas), and $\mathcal{O}$ to denote the set of nominals. We denote elements of $\Sigma$ by letters like $\sigma$, elements of $\mathcal{PROP}$ by letters like $p$ and $q$, and elements of $\mathcal{O}$ by letters like $a$ and $b$.

A Kripke model is a pair $\mathcal{M} = (\Delta^M, \cdot^M)$, where $\Delta^M$ is a set of states and $\cdot^M$ is an interpretation function that maps each nominal $a \in \mathcal{O}$ to an element $a^M$ of $\Delta^M$, each proposition $p \in \mathcal{PROP}$ to a subset $p^M$ of $\Delta^M$ and each atomic program $\sigma \in \Sigma$ to a binary relation $\sigma^M$ on $\Delta^M$. Intuitively, $p^M$ is the set of states in which $p$ is true and $\sigma^M$ is the binary relation consisting of pairs (input state, output state) of the program $\sigma$.

Formulas and programs of the base language of HPDL are defined by the following grammar rules, respectively, where $p \in \mathcal{PROP}$, $a \in \mathcal{O}$ and $\sigma \in \Sigma$:

$$\varphi ::= \top | \bot | p | a | \neg \varphi | \varphi \wedge \varphi | \varphi \vee \varphi | \varphi \rightarrow \varphi | (\alpha)\varphi | [\alpha]\varphi$$

$$\alpha ::= \sigma | \alpha; \beta | \alpha \cup \alpha | \alpha^* | ?\varphi$$

We use letters like $\alpha, \beta$ to denote programs and letters like $\varphi, \psi, \chi$ to denote formulas. The intended meaning of program operators is as follows:

- $\alpha; \beta$ stands for the sequential composition of $\alpha$ and $\beta$,
- $\alpha \cup \beta$ stands for the non-deterministic choice between $\alpha$ and $\beta$,
- $\alpha^*$ stands for the repetition of $\alpha$ a non-deterministic number of times,
- $?\varphi$ stands for checking whether $\varphi$ holds for the current state.

Informally, a formula $(\alpha)\varphi$ represents the set of states $x$ such that the program $\alpha$ has a transition from $x$ to a state $y$ satisfying $\varphi$. Dually, a formula $[\alpha]\varphi$ represents the set of states $x$ from which every transition of $\alpha$ leads to a state satisfying $\varphi$. A formula $a$ (a nominal) represents the set consisting of the only state specified by $a$.

Formally, the interpretation function of a Kripke model $\mathcal{M}$ is extended to interpret complex formulas and complex programs as shown in Fig. 1.

For a set $\Gamma$ of formulas, we denote $\Gamma^M = \bigcap \{ \varphi^M \mid \varphi \in \Gamma \}$. If $w \in \varphi^M$ (resp. $w \in \Gamma^M$), then we say that $\varphi$ (resp. $\Gamma$) is satisfied at $w$ in $\mathcal{M}$. If there exists a Kripke model $\mathcal{M}$ such that $\varphi^M$ (resp. $\Gamma^M$) is not empty, then $\varphi$ (resp. $\Gamma$) is satisfiable.

An assertion is an expression of the form $a : \varphi$ or $\sigma(a, b)$. An ABox is a finite set of assertions. Let $null : \varphi$ stand for $\varphi$. By letters like $a, a_1, a_2$ we will denote nominals or $null$, and by letters like $\xi, \xi'$ we will denote formulas or assertions.

We define:

$$\mathcal{M} \models a : \varphi \quad \text{iff} \quad a^M \in \varphi^M,$$

$$\mathcal{M} \models \sigma(a, b) \quad \text{iff} \quad (a^M, b^M) \in \sigma^M.$$ 

If $\mathcal{M} \models \xi$, then we say that $\mathcal{M}$ satisfies $\xi$. We say that $\mathcal{M}$ satisfies and is a model of an ABox $\Gamma$, and $\Gamma$ is satisfied in $\mathcal{M}$, denoted by $\mathcal{M} \models \Gamma$, if $\mathcal{M}$ satisfies all assertions in $\Gamma$. If $\Gamma$ is satisfied in some Kripke model $\mathcal{M}$, then it is satisfiable.
Formulas $\varphi$ and $\psi$ are equivalent, denoted by $\varphi \equiv \psi$, if $\varphi^M = \psi^M$ for every Kripke model $M$. Assertions $\xi$ and $\zeta$ are equivalent, denoted $\xi \equiv \zeta$, if for every Kripke model $M$, $M \models \xi$ iff $M \models \zeta$.

A formula/assertion is in negation normal form (NNF) if it does not use $\rightarrow$ and it uses $\neg$ only immediately before propositions or nominals. Every formula/assertion can be translated in polynomial time to an equivalent formula/assertion in NNF. From now on, by $\bar{\varphi}$ we denote the NNF of $\neg \varphi$. For an assertion $\xi = a : \varphi$, by $\bar{\xi}$ we denote $a : \bar{\varphi}$. An ABox is in NNF if all of its assertions are in NNF.

### 2.2 Automaton-Modal Operators

We will operate over finite automata instead of regular expressions. Having a formula of the form $[\alpha] \varphi$ (resp. $\langle \alpha \rangle \varphi$), where $\alpha$ is neither an atomic program nor a test, our decision procedure first translates it to $[A] \varphi$ (resp. $\langle A \rangle \varphi$), where $A$ is a finite automaton corresponding to $\alpha$. The operators $[A]$ and $\langle A \rangle$ are called automaton-modal operators and have been used in other works (e.g., [18, 24]). Checking fulfillment of an eventuality $[\alpha] \varphi$ (where $\alpha$ may use the $\ast$ operator) is reduced to checking $\Diamond$-realizability of the corresponding formula $\langle A \rangle \varphi$.

The alphabet $\Sigma(\alpha)$ of a program $\alpha$ and the regular language $L(\alpha)$ generated by $\alpha$ are specified as follows:\footnote{Note that $\Sigma(\alpha)$ contains not only atomic programs but also expressions of the form $\langle \varphi \rangle$, and a program $\alpha$ is a regular expression over its alphabet $\Sigma(\alpha)$.}:

\[
\begin{align*}
\Sigma(\alpha) &= \{ \sigma \} \\
\Sigma(\varphi?) &= \{ \varphi? \} \\
\Sigma(\beta; \gamma) &= \Sigma(\beta) \cup \Sigma(\gamma) \\
\Sigma(\beta \cup \gamma) &= \Sigma(\beta) \cup \Sigma(\gamma) \\
\Sigma(\beta^*) &= \Sigma(\beta) \\
L(\sigma) &= \{ \sigma \} \\
L(\varphi?) &= \{ \varphi? \} \\
L(\beta; \gamma) &= L(\beta) \cdot L(\gamma) \\
L(\beta \cup \gamma) &= L(\beta) \cup L(\gamma) \\
L(\beta^*) &= (L(\beta))^*
\end{align*}
\]

\begin{figure}[h]
\centering
\begin{equation}
(\alpha; \beta)^M = \alpha^M \circ \beta^M \\
(\alpha^*)^M = (\alpha^M)^* \quad (\alpha \cup \beta)^M = \alpha^M \cup \beta^M \\
(\varphi?)^M = \{ (x, x) \mid x \in \varphi^M \} \\
\top^M = \Delta^M \quad \bot^M = \emptyset \\
(\neg \varphi)^M = \Delta^M \setminus \varphi^M \\
(\varphi \land \psi)^M = \varphi^M \cap \psi^M \\
(\alpha \varphi)^M = \{ x \in \Delta^M \mid \exists y((x, y) \in \alpha^M \land y \in \varphi^M) \} \\
(\alpha \varphi)^M = \{ x \in \Delta^M \mid \forall y((x, y) \in \alpha^M \rightarrow y \in \varphi^M) \}
\end{equation}
\caption{Interpretation of complex programs and complex formulas. Here, $\alpha^M \circ \beta^M$ denotes the composition of the binary relations $\alpha^M$ and $\beta^M$, and $(\alpha^M)^*$ the reflexive-transitive closure of $\alpha^M$. In the definition $a^M = \{ a^M \}$, the first $a$ is a formula, while the second $a$ is a nominal. Formally, the formula constructed from a nominal $a$ should be denoted by $\{ a \}$ instead of $a$, and we would have $(\{ a \})^M = \{ a^M \}$. We write $a$ instead of $\{ a \}$ to denote a formula in order to simplify the presentation.}
\end{figure}
Recall that a finite automaton $A$ over an alphabet $\Sigma(\alpha)$ is a tuple $(\Sigma(\alpha), Q, I, \delta, F)$, where $Q$ is a finite set of states, $I \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma(\alpha) \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. A run of $A$ on a word $\omega_1 \ldots \omega_k$ is a finite sequence of states $q_0, q_1, \ldots, q_k$ such that $q_0 \in I$ and $\delta(q_{i-1}, \omega_i, q_i)$ holds for every $1 \leq i \leq k$. It is an accepting run if $q_k \in F$. We say that $A$ accepts a word $w$ if there exists an accepting run of $A$ on $w$. The set of words accepted by $A$ is denoted by $L(A)$.

We will use the following convention:

- Given a finite automaton $A$, we always assume that
  $$A = (\Sigma_A, Q_A, I_A, \delta_A, F_A).$$

- For $q \in Q_A$, we define $\delta_A(q) = \{(\omega, q') \mid (q, \omega, q') \in \delta_A\}$.

As a finite automaton $A$ over an alphabet $\Sigma(\alpha)$ corresponds to a program (the regular expression recognizing the same language), it is interpreted in a Kripke model $M$ as follows:

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

For each program $\alpha$, let $\mathcal{A}_\alpha$ be a finite automaton recognizing the regular language $L(\alpha)$ with the property that $\Sigma_{\mathcal{A}_\alpha} = \Sigma(\alpha)$. The automaton $\mathcal{A}_\alpha$ can be constructed from $\alpha$ in polynomial time. We extend the base language with the auxiliary modal operators $[A,q]$ and $\langle A,q \rangle$, where $A$ is $\mathcal{A}_\alpha$ for some program $\alpha$ and $q$ is a state of $A$. Here, $[A,q]$ and $\langle A,q \rangle$ stand respectively for $[(A,q)]$ and $\langle(A,q)\rangle$, where $(A,q)$ is the automaton that differs from $A$ only in that $q$ is its only initial state. We call $[A,q]$ (resp. $\langle A,q \rangle$) a universal (resp. existential) automaton-modal operator.

In the extended language of HPDL, if $\varphi$ is a formula, then $[A, q] \varphi$ and $\langle A, q \rangle \varphi$ are also formulas. The semantics of these formulas are defined as usual, treating $\langle A, q \rangle$ as a program with semantics specified by (1). From now on, the extended language is used instead of the base language.

Given a Kripke model $M$ and a state $x \in \Delta^M$, we have $x \in ([A,q] \varphi)^M$ (resp. $x \in (\langle A,q \rangle \varphi)^M$) iff

$$x_k \in \varphi^M \text{ for all (resp. some) } x_k \in \Delta^M \text{ such that there exist a word } \omega_1 \ldots \omega_k \text{ (with } k \geq 0\text{) accepted by } (A,q) \text{ with } (x,x_k) \in (\omega_1; \ldots; \omega_k)^M.$$ 

The condition $(x, x_k) \in (\omega_1; \ldots; \omega_k)^M$ means there exist states $x_0 = x, x_1, \ldots, x_{k-1}$ of $M$ such that, for each $1 \leq i \leq k$, if $\omega_i \in \Sigma$ then $(x_{i-1}, x_i) \in \omega_i^M$, else $\omega_i = (\psi_i ?)$ for some $\psi_i$ and $x_{i-1} = x_i$ and $x_i \in \psi_i^M$. Clearly, $\langle A, q \rangle$ is dual to $[A, q]$ in the sense that $\langle A, q \rangle \varphi \equiv \neg [A, q] \neg \varphi$ for any formula $\varphi$.

### 3 A Tableau Calculus for HPDL

From now on, let $\Gamma$ be an ABox in NNF. In this section, we present a tableau calculus $C_{HPDL}$ for checking whether $\Gamma$ is satisfiable. We specify the data structure, the tableau rules, the corresponding tableau decision procedure and state its properties.

#### 3.1 The Data Structure

Let $EdgeLabels$ be the set of formulas and assertions of the form $\langle \sigma \rangle \varphi$ or $a : \langle \sigma \rangle \varphi$. Let the null value indicate a lack of a value as in C programming and SQL.
Definition 1 A tableau is a rooted graph $G = (V, E, v)$, where $V$ is a set of nodes, $E \subseteq V \times V$ is a set of edges, $v \in V$ is the root, each node $v \in V$ has a number of attributes, which form the contents of $v$, and each edge $(v, w)$ may be labeled by a set $\text{ELabels}(v, w) \subseteq \text{EdgeLabels}$. (The contents of nodes and the labels of edges are understood as parts of the graph.) The attributes of a node $v$ are:

- $\text{Type}(v) \in \{\text{state}, \text{non-state}\}$,
- $\text{SType}(v) \in \{\text{complex}, \text{simple}\}$, called the subtype of $v$,
- $\text{Label}(v)$, which is a finite set of assertions or formulas, called the label of $v$,
- $\text{Reduced}(v)$, which is a finite set of so called reduced assertions or formulas of $v$,
- $\text{Status}(v) \in \{\text{unexpanded}, \text{expanded}, \text{incomplete}, \text{blocked}, \text{closed}\} \cup \{\text{closed-wrt}(U) \mid U \subseteq V\}$ and, for all $u \in U$, $\text{Type}(u) = \text{state}$ and $\text{SType}(u) = \text{complex}$,
- $\text{AssSugByNom}(v)$, which is a finite set of so called assertions suggested by nominals for $v$, available (i.e., $\neq \text{null}$) only when $\text{SType}(v) = \text{complex}$ and $\text{Type}(v) = \text{state}$, and is non-empty only when $\text{Status}(v) = \text{incomplete}$,
- $\text{NomRepl}(v) : O \rightarrow O$, which is a partial mapping specifying replacements of nominals for $v$, called in short the nominal replacement for $v$, and is available (i.e., $\neq \text{null}$) only when $\text{SType}(v) = \text{complex}$.

\[\square\]

We define

- $\text{FullLabel}(v) = \text{Label}(v) \cup \text{Reduced}(v)$ if $\text{SType}(v) = \text{simple},$
- $\text{FullLabel}(v) = \text{Label}(v) \cup \text{Reduced}(v) \cup \{a : b \mid \text{NomRepl}(v)(b) = a\}$ otherwise.

In what follows, we define some notions, explain the data structure of tableaux, and present expected properties of tableaux, which will be guaranteed by our construction of tableaux.

If $(v, w) \in E$, then we call $v$ a predecessor of $w$ and $w$ a successor of $v$. Let the relation “being an ancestor” be the reflexive-transitive closure of the relation “being a predecessor”. We say that $v$ is a descendant of $u$ if $u$ is an ancestor of $v$.

We call $v$ a state if $\text{Type}(v) = \text{state}$, and a non-state otherwise. A state is like an “and”-node and a non-state is like an “or”-node, when treating a tableau as an “and-or” graph. States are expanded by using the so-called transitional rule, while non-states are expanded by using other rules. Successors of a state are non-states. Thus, predecessors of a state are non-states. (When a non-state is ready to become a state, we will apply an appropriate expansion rule to connect it to a unique successor that is a state.)

A node $v$ is called a complex node if $\text{SType}(v) = \text{complex}$, and a simple node otherwise. The label of a complex node consists of assertions, while the label of a simple node consists of formulas. Using the terminology of description logic, a complex node is like an ABox consisting of assertions about named individuals, while a simple node is like an unnamed individual and its label consists of properties of that individual. The root $v$ is a complex non-state. If the tableau is intended for checking whether $\Gamma$ is satisfiable, then $\text{Label}(v) = \Gamma$.

The assertions/formulas in the label of a node $v$ are treated as requirements to be realized for $v$. Realizing such requirements causes the graph to be expanded or modified.

A tableau consists of two layers: the layer of complex nodes and the layer of simple nodes. Successors of a simple node are simple nodes, and predecessors of a complex node are complex nodes. The connections between the two layers are edges from complex states to simple non-states.

An edge departing from a node $v$ is labeled if and only if $v$ is a state. For example, if $v$ is a state with $o : \langle \sigma \rangle \varphi \in \text{Label}(v)$, then to realize the requirement $o : \langle \sigma \rangle \varphi$ for $v$, we connect it
to a simple non-state $w$ with $\text{Label}(w) = \{\varphi\} \cup \{\psi \mid o : [\sigma] \psi \in \text{Label}(v)\}$. If the edge $(v, w)$ is established the first time, then it is labeled by $\{o : [\sigma] \varphi\}$. Otherwise, we just add $o : [\sigma] \varphi$ to the labeled of the edge.

For the intuition behind $\text{Reduced}(v)$, consider an example situation when $v$ is an unexpanded non-state, $\varphi \in \text{Label}(v)$ and $\varphi = \psi_1 \vee \psi_2$. To realize the requirement $\varphi$ for $v$, we connect $v$ to two successors $w_1$ and $w_2$ that differ from $v$ in that $\text{Label}(w_i) = \text{Label}(v) \setminus \{\varphi\} \cup \{\psi_i\}$ and $\text{Reduced}(w_i) = \text{Reduced}(v) \cup \{\varphi\}$, for $i \in \{1, 2\}$. In general, $\text{Reduced}(v)$ contains assertions or formulas that have been reduced for $v$.

$\text{Status}(v)$ is called the status of $v$. Possible statuses of nodes are: unexpanded, expanded, incomplete, blocked, closed and closed-wrt($U$), where $U$ is a set of complex states and closed-wrt($U$) is read as “closed w.r.t. any node from $U$”. Informally, closed means “unsatisfiable” and closed-wrt($U$) means “unsatisfiable w.r.t. any node from $U$”. Specifically, if $\text{Status}(v) = \text{closed-wrt}(U)$ and $u \in U$, then: $u$ is an ancestor of $v$ and, if $v = u$, then $\text{FullLabel}(v)$ is unsatisfiable, else $\text{SType}(v) = \text{simple}$ and there does not exist any Kripke model $\mathcal{M}$ such that $\mathcal{M} \models \text{FullLabel}(u)$ and $(\text{Label}(v))^\mathcal{M} \neq \emptyset$. Clearly, closed-wrt($U$) does not mean closed as it requires “contexts”. By closed-wrt(…$)$ we denote closed-wrt($U$) for some $U$, and by closed-wrt$_1$($u$) we denote closed-wrt($U$) for some $U$ containing $u$. When negated, e.g., in the form $\not\equiv \text{closed-wrt}_1$($u$) or $\not\in \{\text{closed-wrt}_1$($u$), …$\}$, we mean the considered status is different from closed-wrt($U$) for any $U$ that contains $u$. A node $v$ may have status incomplete only when it is a complex state, and this status means that we would like to extend the label of $v$ with the assertions from $\text{AssSugByNom}(v)$ as one of the possibilities. A node may have status blocked only when it is a simple node whose label contains some nominals. The status blocked can be updated only to closed or closed-wrt(…$)$.

**Remark 1** Let us explain how we will deal with nominals.

If $v$ is an unexpanded complex non-state with $a : b \in \text{Label}(v)$, where $a \neq b$, then we can connect $v$ to a complex non-state $w$, whose contents are obtained from the contents of $v$ by replacing every occurrence of $b$ by $a$, and then set $\text{NomRepl}(w)(b) := a$. The rule for doing this will be formally specified later.

Complex nodes in a tableau are possible expansions of the root. The aim is to check whether the tableau has a complex node $u$ such that $\text{FullLabel}(u)$ is satisfiable, and one can restrict to the case when $u$ is a state. Let $u$ be a complex state. It contains information (requirements) about nominals. Let $v$ be a simple node and a descendant of $u$ such that $\text{Label}(v)$ contains a nominal $a$. Thus, $v$ should represent the state named $a$ in the intended Kripke model. It should be “compatible” with the information about $a$ specified by $\text{FullLabel}(u)$.

Let $X = \{a : \varphi \mid \varphi \in \text{Label}(v) \land \varphi \neq a\}$.

If there exists $\xi \in X$ such that $\xi \in \text{FullLabel}(u)$, then there is a clash between $v$ and $u$, which means that the expansion paths from $u$ to $v$ cannot be used and we can set the status of $v$ to closed-wrt$_1$($u$).  

Otherwise, if $X \not\subseteq \text{FullLabel}(u)$, then, from the viewpoint of $v$, the assertions from $X \setminus \text{FullLabel}(u)$ should be added to the label of $u$. Note, however, that the expansion paths from $u$ to $v$ may contain “or”-branchings and what $v$ requires from $u$ is just one of possibilities for $u$ and can be treated as a “suggestion”. So, in this case, we set $\text{Status}(u) := \text{incomplete}$ and $\text{AssSugByNom}(u) := X \setminus \text{FullLabel}(u)$.

We treat $v$ just as a secondary appearance of the nominal $a$ specified by $\text{FullLabel}(u)$. What further should be done for $a$ (and $v$) is a matter of complex states like $u$. So, after the above mentioned changes, if $\text{Status}(v)$ is still unexpanded, then we set $\text{Status}(v) := \text{blocked}$.

---

3 That is, if $\text{Status}(v) \notin \{\text{closed}, \text{closed-wrt}(…)$, then $\text{Status}(v) := \text{closed-wrt}(\{u\})$, else if $\text{Status}(v) = \text{closed-wrt}(U)$ then $\text{Status}(v) := \text{closed-wrt}(U \cup \{u\})$. 

Springer
node \( v \) will not be expanded, but its status can be changed (either to closed or closed-wrt(\ldots)). This technique aims to ensure the unicity of the interpretation of nominals.

Now, assume that \( \text{Status}(u) = \text{incomplete} \) and let \( x \) be a predecessor of \( u \). As discussed earlier, \( x \) is a complex non-state. By construction, \( x \) and \( u \) have the same attributes \( \text{Label}, \text{Reduced} \) and \( \text{NomRepl} \). Since \( \text{Status}(u) = \text{incomplete} \), from the viewpoint of \( x \), the expansion from \( x \) to \( u \) (for transforming \( x \) to a state) was not appropriate. So, we re-expand \( x \) as follows:

- delete the edge \((x, u)\);
- connect \( x \) to a complex non-state that differs from \( x \) in that its label is \( \text{Label}(x) \cup \text{AssSugByNom}(u) \);
- for each \( \xi \in \text{AssSugByNom}(u) \), connect \( x \) to a complex non-state that differs from \( x \) in that its label is \( \text{Label}(x) \cup \{\xi\} \).

\[ \square \]

**Remark 2** A tableau is constructed with global caching in the sense that, if \( v_1 \) and \( v_2 \) are different nodes, then \( \text{Type}(v_1) \neq \text{Type}(v_2) \) or \( \text{SType}(v_1) \neq \text{SType}(v_2) \) or \( \text{Label}(v_1) \neq \text{Label}(v_2) \) or \( \text{Reduced}(v_1) \neq \text{Reduced}(v_2) \) or \( \text{NomRepl}(v_1) \neq \text{NomRepl}(v_2) \).

Connecting a node \( v \) to a successor, which is created if necessary, is done by Function \( \text{ConToSucc}(v, \text{type}, \text{SType}, \text{label}, \text{reduced}, \text{NomRepl}, \text{eLabel}) \) on page 10, where the parameters \( \text{type}, \text{SType}, \text{label}, \text{reduced}, \text{NomRepl} \) specify the attributes of the successor, and \( \text{eLabel} \) stands for an edge label of the connection. By applying global caching, we first check whether an existing node can be used as such a successor of \( v \). If not, a new node is created and used as a successor of \( v \). The parameter \( \text{eLabel} \) is null iff \( \text{Type}(v) = \text{non-state} \), and the parameter \( \text{NomRepl} \) is null iff the parameter \( \text{SType} \) is simple.

**Function** \( \text{ConToSucc}(v, \text{type}, \text{SType}, \text{label}, \text{reduced}, \text{NomRepl}, \text{eLabel}) \)

- **Global data:** a tableau \((V, E, v)\) in construction.
- **Purpose:** connect a node \( v \) to a successor, which is created if necessary.
- **1** if there exists a node \( w \in V \) such that \( \text{Type}(w) = \text{type}, \text{SType}(w) = \text{SType}, \text{Label}(w) = \text{label}, \text{Reduced}(w) = \text{reduced} \quad \text{and} \quad \text{NomRepl}(w) = \text{NomRepl} \) then
- **2** if \((v, w) \notin E\) then
- **3** \( E := E \cup \{(v, w)\} \);
- **4** if \( \text{Type}(v) = \text{state} \) then \( \text{ELabels}(v, w) := \{\text{eLabel}\} \);
- **5** else if \( \text{Type}(v) = \text{state} \) then add \( \text{eLabel} \) to \( \text{ELabels}(v, w) \);
- **6** else
- **7** add a new node \( w \) to \( V \);
- **8** \( E := E \cup \{(v, w)\} \);
- **9** if \( \text{Type}(v) = \text{state} \) then \( \text{ELabels}(v, w) := \{\text{eLabel}\} \);
- **10** \( \text{Type}(w) := \text{type}, \text{SType}(w) := \text{SType}, \text{Status}(w) := \text{unexpanded} \);
- **11** \( \text{Label}(w) := \text{label}, \text{Reduced}(w) := \text{reduced}, \text{NomRepl}(w) := \text{nomRepl} \);
- **12** if \( \text{SType} = \text{complex} \quad \text{and} \quad \text{type} = \text{state} \) then \( \text{AssSugByNom}(w) := \emptyset \);
- **13** return \( w \);

### 3.2 Tableau Rules

Our tableau calculus \( C_{\text{HPDL}} \) consists of the following tableau rules:

- The static rules for expanding a non-state,
The rule \((\text{repl-nom})\) for replacing nominals in a complex non-state,

The rule \((\text{nominal})\) for dealing with nominals in a simple non-state,

The rule \((\text{re-expand})\) for re-expanding a complex non-state,

The rule \((\text{form-state})\) for forming a state,

The transitional rule \((\text{trans})\) for expanding a state,

The rules \((\text{close}_1)-(\text{close}_4)\) for updating the status of a node to \(\text{closed}\) or \(\text{closed-wrt}(\ldots)\).

The applicability of a rule to a tableau is explicitly specified for the static rules. For any of the other rules, we say that it is \(\text{applicable}\) to a tableau if its execution can make changes to the tableau.

### 3.2.1 The Static Rules for Expanding a Non-State

The static rules are written downwards, with a set of assertions/formulas above the line as the \textit{premise}, which represents the label of the node to which the rule is applied, and a number of sets of assertions/formulas below the line as the \textit{(possible) conclusions}, which represent the labels of the successor nodes resulted from the application of the rule. Possible conclusions of a static rule are separated by \(|\). If a rule is unary (i.e., with only one possible conclusion), then its only conclusion is “firm” and we ignore the word “possible”. The meaning of a static rule is that, if the premise is satisfiable, then some of the possible conclusions are also satisfiable.

We use \(X\) and \(Y\) to denote sets of assertions/formulas and write \(X, o: \varphi\) to denote \(X \cup \{o: \varphi\}\) with the assumption that \(o: \varphi \notin X\). The static rules of \(C_{\text{HPDL}}\) are specified in Table \(\text{1}\) as schemas. For each of them, the distinguished assertions/formulas of the premise are called the \textit{principal assertions/formulas} of the rule. A static rule \((\rho)\) as an instance of a schema \((\text{given in Table 1})\) is \(\text{applicable}\) to a node \(v\) if the following conditions hold:

\begin{itemize}
  \item \(\text{Status}(v) = \text{unexpanded}\) and \(\text{Type}(v) = \text{non-state}\).
  \item \(\text{The rules (repl-nom) and (nominal) are not applicable to } v\).
  \item \(\text{The premise of the rule is equal to Label}(v)\).
  \item \(\text{The conditions accompanied with } (\rho) \text{ are satisfied}\).
  \item \(\text{If } (\rho) \neq (\Box \text{trans}) \text{, then the principal assertion/formula of } (\rho) \text{ does not belong to Reduced}(v), \text{else the assertion } b: \varphi \text{ in the conclusion does not belong to FullLabel}(v)\).
\end{itemize}

The last condition prevents applying the rule unnecessarily, because it has been applied to an ancestor node of \(v\) that corresponds to the same state in the intended Kripke model.

If \((\rho) \neq (\Box \text{trans})\) is a static rule applicable to \(v\), then the application is as follows:

\begin{itemize}
  \item Let \(\xi\) be the principal assertion/formula of \((\rho)\).
  \item Let \(X_1, \ldots, X_k\) be the possible conclusions of \((\rho)\).
  \item For each \(1 \leq i \leq k\), do \(\text{ConToSucc}(v, \text{non-state}, \text{SType}(v), X_i, \text{Reduced}(v) \cup \{\xi\}, \text{NomRepl}(v), \text{null})\), which is specified on page \(10\).
  \item \(\text{Status}(v) := \text{expanded}\).
\end{itemize}

If \((\Box \text{trans})\) is applicable to \(v\), then the application is as follows:

\begin{itemize}
  \item Let \(Y\) be the conclusion of \((\Box \text{trans})\).
  \item \(\text{ConToSucc}(v, \text{non-state}, \text{SType}(v), Y, \text{Reduced}(v), \text{NomRepl}(v), \text{null})\).
  \item \(\text{Status}(v) := \text{expanded}\).
\end{itemize}

Applying a static rule understood as a schema to a node \(v\) means applying an instance of the schema to \(v\). Such an instance is chosen as follows: choose assertions/formulas from
Table 1 The static rules of $C_{\text{HPDL}}$

| Rule | Premise | Conclusion |
|------|---------|------------|
| $(\land)$ | $X, o:(\varphi \land \psi)$ | $X, o:\varphi, o:\psi$ |
| $(\lor)$ | $X, o:(\varphi \lor \psi)$ | $X, o:\varphi \lor X, o:\psi$ |

if $\alpha \notin \Sigma$, $\alpha$ is not a test, and $I_{\alpha} = \{q_1, \ldots, q_k\}$:

| Rule | Premise | Conclusion |
|------|---------|------------|
| $(\text{aut}_\square)$ | $X, o:[\alpha]\varphi$ | $X, o:[\alpha, q_1]\varphi, \ldots, o:[\alpha, q_k]\varphi$ |
| $(\text{aut}_\diamond)$ | $X, o:(\alpha)\varphi$ | $X, o:(\alpha, q_1)\varphi \lor \ldots \lor X, o:(\alpha, q_k)\varphi$ |

if $\delta_A(q) = \{ (\omega_1, q_1), \ldots, (\omega_k, q_k) \}$ and $q \notin F_A$:

| Rule | Premise | Conclusion |
|------|---------|------------|
| $(\text{[A]})$ | $X, o:[A, q]\varphi$ | $X, o:[\omega_1][A, q_1]\varphi, \ldots, o:[\omega_k][A, q_k]\varphi$ |
| $(\text{\{A\}})$ | $X, o:(A, q)\varphi$ | $X, o:(\omega_1)(A, q_1)\varphi \lor \ldots \lor X, o:(\omega_k)(A, q_k)\varphi$ |

if $\delta_A(q) = \{ (\omega_1, q_1), \ldots, (\omega_k, q_k) \}$ and $q \in F_A$:

| Rule | Premise | Conclusion |
|------|---------|------------|
| $(\text{\{A\}_f})$ | $X, o:[A, q]\varphi$ | $X, o:[\omega_1][A, q_1]\varphi, \ldots, o:[\omega_k][A, q_k]\varphi, o:\varphi$ |
| $(\text{\{A\}}_f)$ | $X, o:(A, q)\varphi$ | $X, o:(\omega_1)(A, q_1)\varphi \lor \ldots \lor X, o:(\omega_k)(A, q_k)\varphi \lor X, o:\varphi$ |

| Rule | Premise | Conclusion |
|------|---------|------------|
| $(\square ?)$ | $X, o:[\psi]?\varphi$ | $X, o:\overline{\psi} \lor X, o:\varphi$ |
| $(\square ?)$ | $X, o:(\psi)\varphi$ | $X, o:\psi, o:\varphi$ |
| $(\square_{\text{trans}})$ | $X, a:[\sigma]\varphi, \sigma(a, b)$ | $X, a:[\sigma]\varphi, \sigma(a, b), b:\varphi$ |

Label($v$) such that they can be “unified” with the principal assertions/formulas in the schema, then instantiate the schema by using the substitution resulted from that unification.

For explanations of the rules (repl-nom), (nominal) and (re-expand) specified below, the reader can recall Remark 1.

3.2.2 The Rule (repl-nom) for Replacing Nominals

If Status($v$) = unexpanded and Label($v$) contains $a:b$ with $a \neq b$ then:

1. let $X$ and $Y$ be the sets obtained from Label($v$) \ {$a:b$} and Reduced($v$), respectively, by replacing every occurrence of $b$ with $a$, including the ones in automata of modal operators;
2. $w := \text{ConToSucc}(v, \text{non-state, complex, } X, Y, \text{NomRepl}(v), \text{null})$;
3. NomRepl($w$)($b$) := $a$;
4. for each nominal $c$ such that NomRepl($v$)($c$) = $b$, do NomRepl($w$)($c$) := $a$;
5. Status($v$) := expanded;
### 3.2.3 The Rule (nominal) for Dealing with Nominals

If $\text{Status}(v) \neq \text{closed}$, $\text{Type}(v) = \text{simple}$ and there exists $a \in \text{Label}(v)$, then:

1. for each complex state $u$ such that $\text{Status}(v) \neq \text{closed-wrt}_+(u)$, $\text{Status}(u) \neq \text{incomplete}$ and there exists a path from the root $v$ to $u$ via $u$ that does not contain any node with status $\text{closed}$ or $\text{closed-wrt}_+(u)$, do:
   
   (a) $X := \{a : \varphi \mid \varphi \in \text{Label}(v) \text{ and } \varphi \neq a\}$;
   
   (b) if there exists $\xi \in X$ such that $\xi \in \text{FullLabel}(u)$ then
      
      i. if $\text{Status}(v)$ is of the form $\text{closed-wrt}(U)$ then $\text{Status}(v) := \text{closed-wrt}(U \cup \{u\})$,
      
      ii. else $\text{Status}(v) := \text{closed-wrt}(\{u\})$;
   
   (c) else if $X \not\subseteq \text{FullLabel}(u)$ then
      
      i. $\text{Status}(u) := \text{incomplete}$;
      
      ii. $\text{AssSugByNom}(u) := X - \text{FullLabel}(u)$;

2. if $\text{Status}(v) = \text{unexpanded}$ then $\text{Status}(v) := \text{blocked}$.

### 3.2.4 The Rule (re-expand) for Re-expanding a Complex Non-state

If $(v, w) \in E$ and $\text{Status}(w) = \text{incomplete}$ then:

(we must have that $\text{SType}(v) = \text{complex}$ and $\text{Type}(v) = \text{non-state}$)

1. delete the edge $(v, w)$ from $E$;
2. $X := \text{Label}(v) \cup \text{AssSugByNom}(w)$;
3. $\text{ConToSucc}(v, \text{non-state}, \text{SType}(v), X, \text{Reduced}(v), \text{NomRepl}(v), \text{null})$;
4. for each $\xi \in \text{AssSugByNom}(w)$, do $\text{ConToSucc}(v, \text{non-state}, \text{SType}(v), \text{Label}(v) \cup \{\xi\}, \text{Reduced}(v), \text{NomRepl}(v), \text{null})$.

### 3.2.5 The Rule (form-state) for Forming a State

If $\text{Status}(v) = \text{unexpanded}$, $\text{Type}(v) = \text{non-state}$ and no rule among the static rules, (repl-nom) and (nominal) is applicable to $v$, then:

1. if $\text{SType}(v) = \text{complex}$ then $\text{ConToSucc}(v, \text{state}, \text{complex}, \text{Label}(v), \text{Reduced}(v), \text{NomRepl}(v), \text{null})$,
2. else $\text{ConToSucc}(v, \text{state}, \text{simple}, \text{Label}(v), \text{Reduced}(v), \text{null}, \text{null})$;
3. $\text{Status}(v) := \text{expanded}$.

### 3.2.6 The Transitional Rule (trans) for Expanding a State

If $\text{Type}(v) = \text{state}$ and $\text{Status}(v) = \text{unexpanded}$ then:

1. for each $o : (\sigma) \varphi \in \text{Label}(v)$ do
   
   (a) $X := \{\varphi\} \cup \{\psi \mid o : [\sigma] \psi \in \text{Label}(v)\}$;
   
   (b) $\text{ConToSucc}(v, \text{non-state}, \text{simple}, X, \emptyset, \text{null}, o : (\sigma) \varphi)$;

2. $\text{Status}(v) := \text{expanded}$.
3.2.7 The Rules (close\(_1\))–(close\(_4\)) for Updating the Status of a Node

We need the following definition before specifying the rules (close\(_1\))–(close\(_4\)).

**Definition 2** Let \(\xi \in FullLabel(v)\) be of the form \(o' : \langle A, q'\rangle \varphi\) or \(o' : \langle \omega'\rangle \langle A, q'\rangle\varphi\). We say that \(\xi\) is \(\diamond\)-realizable at \(u\) w.r.t. \(v\), where \(u\) is a complex state and an ancestor of \(v\) such that

1. \(v_0 = v, \xi_0 = \xi\) and, for \(1 \leq i \leq h, v_i \in V\) and \(\xi_i \in FullLabel(v_i)\);
2. for \(0 \leq i < h\), \(\text{Status}(v_i) \notin \{\text{closed, incomplete}\}\); if there exists a sequence \((v_0, \xi_0), \ldots, (v_h, \xi_h)\) such that:
   1. \(v_0 = v, \xi_0 = \xi\) and, for \(1 \leq i \leq h, v_i \in V\) and \(\xi_i \in FullLabel(v_i)\);
   2. for \(0 \leq i < h\), \(\text{Status}(v_i) \notin \{\text{closed, closed-wrt}\_\_\_ (u)\}\);
   3. \(\text{Status}(v_h) = \text{unexpanded}\) or there exists \(o\) such that \(\xi_h = o' : \varphi\);
   4. for each \(0 \leq i < h\), there exist \(o, q, q'', \omega, \psi, \sigma\) and/or \(a\) such that some of the following conditions hold:

   (a) \(i = h - 1\), \(\xi_i = o' : \langle A, q\rangle \varphi\), \(q \in F\_A\), \(v_{i+1} = v_i\) and \(\xi_{i+1} = o' : \varphi\);
   (b) \(\xi_i = o' : \langle A, q\rangle \varphi, \langle q, \omega, q''\rangle \in \delta\_A\), \(v_{i+1} = v_i\) and \(\xi_{i+1} = o' : \langle \omega\rangle \langle A, q''\rangle \varphi\);
   (c) \(\xi_i = o' : \langle \psi'\rangle \langle A, q\rangle \varphi, o' : \psi \in FullLabel(v_i), v_{i+1} = v_i\) and \(\xi_{i+1} = o' : \langle A, q\rangle \varphi\);
   (d) \(i = h - 1\), \(v_i\) was expanded by the tableau rule \((\langle A\rangle f)\) with \(\xi_i = o' : \langle A, q\rangle \varphi\) as the principal assertion/formula, \(v_{i+1}\) is the successor of \(v_i\) such that \(Label(v_{i+1})\) is obtained from \(Label(v_i)\) by replacing \(\xi_i\) with \(\xi_{i+1} = o' : \varphi\);
   (e) \(v_i\) was expanded by the tableau rule \((\langle A\rangle)\) or \((\langle A\rangle f)\) with \(\xi_i = o' : \langle A, q\rangle \varphi\) as the principal assertion/formula, \(v_{i+1}\) is a successor of \(v_i\) and \(\xi_{i+1} = o' : \langle \omega\rangle \langle A, q''\rangle \varphi\) is the assertion/formula obtained from \(\xi_i\);
   (f) \(v_i\) was expanded by the tableau rule \((\diamond \?)\) with \(\xi_i = o' : \langle \psi'\rangle \langle A, q\rangle \varphi\) as the principal assertion/formula, \(v_{i+1}\) is the unique successor of \(v_i\) and \(\xi_{i+1} = o' : \langle A, q\rangle \varphi\);
   (g) \(v_i\) was expanded by the rule \((\text{form-state})\) or a static tableau rule and \(\xi_i\) is not a principal assertion/formula, \(v_{i+1}\) is a successor of \(v_i\) and \(\xi_{i+1} = \xi_i\);
   (h) \(v_i\) was expanded by the rule \((\text{trans})\), \(\xi_i = o' : \langle \sigma\rangle \langle A, q\rangle \varphi\), \(v_{i+1}\) is a successor of \(v_i\), \(\xi_i \in E\_Labels(v_i, v_{i+1})\) and \(\xi_{i+1} = o' : \langle A, q\rangle \varphi\);
   (i) \(a \in Label(v_i)\), \(v_{i+1} = u\) and \(\xi_{i+1} = a : \xi_i\).

If \(\text{Status}(v_h) \neq \text{unexpanded}\), then the sequence \((v_0, \xi_0), \ldots, (v_h, \xi_h)\) is called a \(\diamond\)-realization of \(\xi\) at \(u\) w.r.t. \(v\).

Note that checking \(\diamond\)-realizability can be done in a similar way as for checking productivity of states in a finite automaton.

The rules (close\(_1\))–(close\(_4\)) are specified as follows:

\textbf{(close\(_1\))}:

If \(\text{Status}(v) \neq \text{closed}\) and either (there exists \(o : \bot \in Label(v)\) or \(a : \neg a \in Label(v)\) or \(\{\xi, \overline{\xi}\} \subseteq FullLabel(v)\) or \(\text{Status}(v) = closed-wrt_\_\_ (v)\)), then:

\[\text{Status}(v) := \text{closed}.\]

\textbf{(close\(_2\))}:

If \(\text{Status}(v) \neq \text{closed}\) and \(u\) is a complex state and an ancestor of \(v\) such that \(\text{Status}(u) \neq \{\text{closed, incomplete}\}\) and there exists \(\xi \in FullLabel(v)\) of the form \(o : \langle A, q\rangle \varphi\) or \(o : \langle \omega\rangle \langle A, q\rangle \varphi\) that is not \(\diamond\)-realizable at \(v\) w.r.t. \(u\) and \(\text{Status}(v) \neq closed-wrt_\_\_ (u)\), then:

\[\text{closed}\]
1. if Status(v) is of the form \textit{closed}\text{-wrt}(U), then Status(v) := \textit{closed}\text{-wrt}(U \cup \{u\})
2. else Status(v) := \textit{closed}\text{-wrt(\{u\})}

\textit{(close3):}

If Status(v) \notin \{\text{unexpanded}, \text{closed}, \text{blocked}, \text{closed}\text{-wrt(\ldots)}\} and Type(v) = \text{non-state}, then:
1. if all successors of v have status \textit{closed} then Status(v) := \textit{closed},
2. else if every successor of v has status \textit{closed} or \textit{closed}\text{-wrt(\ldots)} then:
   \begin{enumerate}
   \item let w_1, \ldots, w_k be all the successors of v such that, for 1 \leq i \leq k, Status(w_i) is of the form \textit{closed}\text{-wrt}(U_i), and let \( U = \bigcap_{1 \leq i \leq k} U_i \);
   \item if \( U \neq \emptyset \) then: if Status(v) is of the form \textit{closed}\text{-wrt}(U'), then Status(v) := \textit{closed}\text{-wrt}(U' \cup U), else Status(v) := \textit{closed}\text{-wrt}(U).
   \end{enumerate}

\textit{(close4):}

If Status(v) \notin \{\text{unexpanded}, \text{closed}, \text{incomplete}\} and Type(v) = \text{state}, then:
1. if v has a successor w with Status(w) = \textit{closed}, then Status(v) := \textit{closed},
2. else if v has a successor w with Status(w) = \textit{closed}\text{-wrt}(U) and Status(v) is not of the form \textit{closed}\text{-wrt}(U') with \( U' \supseteq U \), then:
   \begin{enumerate}
   \item if Status(v) is of the form \textit{closed}\text{-wrt}(U'), then Status(v) := \textit{closed}\text{-wrt}(U' \cup U),
   \item else Status(v) := \textit{closed}\text{-wrt}(U).
   \end{enumerate}

### 3.3 Checking Satisfiability

Let \( \Gamma \) be an ABox in NNF. A C\text{\textscript{HPDL}}-tableau for \( \Gamma \) is a tableau \( G = (V, E, v) \) constructed as follows. At the beginning:

\begin{itemize}
   \item \( V := \{v\}, E := \emptyset \),
   \item Type(v) := \text{non-state}, SType(v) := \text{complex}, Status(v) := \text{unexpanded},
   \item Label(v) := \Gamma, Reduced(v) := \emptyset, NomRepl(v) := \emptyset,
   \item for each nominal a occurring in \( \Gamma \), NomRepl(v)(a) := a.
\end{itemize}

Then, while Status(v) \neq \textit{closed} and there is some tableau rule (\( \rho \)) applicable to some node v, choose such a pair ((\( \rho \)), v) and apply (\( \rho \)) to v.\footnote{As an optimization, it makes sense to expand v only when there may exist a path from the root to v that does not contain any node with the status \textit{closed}.} Observe that the set of all assertions and formulas that may appear in the contents of the nodes of G is finite. Due to global caching, G is finite and can be effectively constructed. The following theorem immediately follows from Corollaries 3 and 4, which are given and proved in Sect. 5.

**Theorem 1** (Soundness and Completeness) Let \( \Gamma \) be an ABox in NNF and G = (V, E, v) an arbitrary C\text{\textscript{HPDL}}-tableau for \( \Gamma \). Then, \( \Gamma \) is satisfiable if and only if Status(v) \neq \textit{closed}. \( \square \)

To check satisfiability of an ABox \( \Gamma \) in NNF, one can construct a C\text{\textscript{HPDL}}-tableau \( G = (V, E, v) \) for \( \Gamma \) and return “no” when Status(v) = \textit{closed}, or “yes” otherwise. We call this the C\text{\textscript{HPDL}}-tableau decision procedure. The following corollary immediately follows from Corollary 2, which is given and proved in Sect. 5.

**Corollary 1** The C\text{\textscript{HPDL}}-tableau decision procedure has the ExpTime complexity. \( \square \)
4 An Illustrative Example

Consider the following ABox in NNF:

$$\Gamma = \{a: [\sigma^*]p, \sigma(a, b), b: (a? \cup \sigma)^* \neg p\}.$$  

We show that $\Gamma$ is unsatisfiable by constructing a $\mathcal{C}_{\text{HPDL}}$-tableau $G = (V, E, \nu)$ for $\Gamma$ with $\text{Status}(\nu) = \text{closed}$. In the construction, we use the following finite automata:

$$A_1 = A_{\sigma^*} = ([\sigma], \{0\}, \{0\}, \{(0, \sigma, 0), \{0\}\}),$$

$$A_2 = A_{(a? \cup \sigma)^*} = ([\sigma, a?], \{0\}, \{0\}, \{(0, \sigma, 0), (0, a?, 0), \{0\}\}),$$

$$A_3 = A_{(b? \cup \sigma)^*} = ([\sigma, b?], \{0\}, \{0\}, \{(0, \sigma, 0), (0, b?, 0), \{0\}\}).$$

As these automata have only one state (named 0), we will write $A_i$ instead of $(A_i, 0)$, for $1 \leq i \leq 3$. For example, $[A_1]$ and $\langle A_1 \rangle$ stand for $[A_1, 0]$ and $\langle A_1, 0 \rangle$, respectively. The constructed $\mathcal{C}_{\text{HPDL}}$-tableau $G$ is illustrated in Fig. 2.

At the beginning, $G$ contains the root $\nu$ with $\text{Label}(\nu) = \Gamma$.

Applying $(\text{aut}_{\Box})$, $\nu$ is connected to a new complex non-state $v_1$ with

$$\text{Label}(v_1) = \{a: [A_1]p, \sigma(a, b), b: (a? \cup \sigma)^* \neg p\}.$$
Applying ([A]_f), \( v_1 \) is connected to a new complex non-state \( v_2 \) with
\[
\text{Label}(v_2) = \{a : \sigma[A_1]p, \ a : p, \ \sigma(a, b), \ b : (a? \cup \sigma)^* \neg p\}.
\]
Applying (\( \square_{\text{trans}} \)), \( v_2 \) is connected to a new complex non-state \( v_3 \) with
\[
\text{Label}(v_3) = \text{Label}(v_2) \cup \{b : [A_1]p\}.
\]
Applying ([A]_f), \( v_3 \) is connected to a new complex non-state \( v_4 \) with
\[
\text{Label}(v_4) = \text{Label}(v_2) \cup \{b : [A_1]p, \ b : p\}.
\]
Applying (\( \text{aut}_{\emptyset} \)), \( v_4 \) is connected to a new complex non-state \( v_5 \) with
\[
\text{Label}(v_5) = \{a : \sigma[A_1]p, \ a : p, \ \sigma(a, b), \ b : [\sigma[A_1]p, \ b : p, \ b : (A_2) \neg p\}.
\]
Applying ([\( \land \)]_f), \( v_5 \) is connected to new complex non-states \( v_6, v_7, \) and \( v_8 \) with
\[
\text{Label}(v_6) = \{a : \sigma[A_1]p, \ a : p, \ \sigma(a, b), \ b : [\sigma[A_1]p, \ b : p, \ b : \neg p\}.
\]
\[
\text{Label}(v_7) = \{a : \sigma[A_1]p, \ a : p, \ \sigma(a, b), \ b : [\sigma[A_1]p, \ b : p, \ b : (\sigma) \langle A_2 \rangle \neg p\}.
\]
\[
\text{Label}(v_8) = \{a : \sigma[A_1]p, \ a : p, \ \sigma(a, b), \ b : [\sigma[A_1]p, \ b : p, \ b : (a?) \langle A_2 \rangle \neg p\}.
\]
Applying (\( \text{close}_1 \)), \( \text{Status}(v_6) \) is changed to \textit{closed}.
Applying (\( \text{form-state} \)), \( v_7 \) is connected to a new complex state \( v_9 \) with \( \text{Label}(v_9) = \text{Label}(v_7) \). Applying (\( \text{trans} \)), \( v_9 \) is connected to a new simple non-state \( v_{10} \) with
\[
\text{Label}(v_{10}) = \{\langle A_2 \rangle \neg p, \ [A_1]p\}.
\]
\[
\text{ELabels}(v_9, v_{10}) = \{b : \langle \sigma \rangle \langle A_2 \rangle \neg p\}.
\]
Applying ([A]_f), \( v_{10} \) is connected to a new simple non-state \( v_{11} \) with
\[
\text{Label}(v_{11}) = \{\langle A_2 \rangle \neg p, \ [\sigma][A_1]p, \ p\}.
\]
Applying ([\( \land \)]_f), \( v_{11} \) is connected to new simple non-states \( v_{12}, v_{13}, \) and \( v_{14} \) with
\[
\text{Label}(v_{12}) = \{\neg p, \ [\sigma][A_1]p, \ p\}.
\]
\[
\text{Label}(v_{13}) = \{\langle \sigma \rangle \langle A_2 \rangle \neg p, \ [\sigma][A_1]p, \ p\}.
\]
\[
\text{Label}(v_{14}) = \{\langle a? \rangle \langle A_2 \rangle \neg p, \ [\sigma][A_1]p, \ p\}.
\]
Applying (\( \text{close}_1 \)), \( \text{Status}(v_{12}) \) is changed to \textit{closed}.
Applying (\( \text{form-state} \)), \( v_{13} \) is connected to a new simple state \( v_{15} \) with \( \text{Label}(v_{15}) = \text{Label}(v_{13}) \). Applying (\( \text{trans} \)), \( v_{15} \) is connected to the existing node \( v_{10} \) with \( \text{ELabels}(v_{15}, v_{10}) = \{\langle \sigma \rangle \langle A_2 \rangle \neg p\} \).
Applying (\( \hat{\diamond} \)), \( v_{14} \) is connected to a new simple non-state \( v_{16} \) with
\[
\text{Label}(v_{16}) = \{a, \ \langle A_2 \rangle \neg p, \ [\sigma][A_1]p, \ p\}.
\]
Applying (\( \text{nominal} \)) to \( v_{16} \) and the complex state \( v_9 \), \( \text{Status}(v_9) \) is changed to \textit{incomplete}, \( \text{AssSugByNom}(v_9) \) is set to \( \{a : \langle A_2 \rangle \neg p\} \), and \( \text{Status}(v_{16}) \) is changed to \textit{blocked}.
Applying (\( \text{re-expand} \)) to \( v_7 \) and the incomplete complex state \( v_9 \), the edge \( (v_7, v_9) \) is deleted and \( v_7 \) is connected to new complex non-states \( v_{17} \) and \( v_{18} \) with
\[
\text{Label}(v_{17}) = \text{Label}(v_7) \cup \{a : \langle A_2 \rangle \neg p\},
\]
\[
\text{Label}(v_{18}) = \text{Label}(v_7) \cup \{a : [A_2]p\}.
\]
Applying (\langle A \rangle_f), v_{17} is connected to new complex non-states \( v_{19}, v_{20}, v_{21} \) with

\[
\text{Label}(v_{19}) = \text{Label}(v_7) \cup \{a : \neg p\},
\]
\[
\text{Label}(v_{20}) = \text{Label}(v_7) \cup \{a : \langle \sigma \rangle \langle A_2 \rangle p\},
\]
\[
\text{Label}(v_{21}) = \text{Label}(v_7) \cup \{a : a ? \langle A_2 \rangle \neg p\}.
\]

Applying (\textit{close}$_1$), \textit{Status}(v_{19}) is changed to \textit{closed} (due to \( a : p\) and \( a : \neg p\)).

Applying (\textit{form-state}), \( v_{20} \) is connected to a new complex state \( v_{22} \) with \textit{Label}(v_{22}) = Label(v_{20}). \text{Applying (trans)}, \( v_{22} \) is connected to the existing node \( v_{10} \) with \( ELables(v_{22}, v_{10}) = \{a : \langle \sigma \rangle \langle A_2 \rangle \neg p, b : \langle \sigma \rangle \langle A_2 \rangle \neg p\}. \)

Observe that the assertion \( b : \langle \sigma \rangle \langle A_2 \rangle \neg p \) is not \( \lozenge \)-realizable at \( v_{22} \) w.r.t. \( v_{22} \). Hence, applying (\textit{close}$_2$) and (\textit{close}$_1$), \textit{Status}(v_{22}) is first changed to \textit{closed-wrt}((v_{22})) and then to \textit{closed}; after that, by applying (\textit{close}$_3$), \textit{Status}(v_{20}) is also changed to \textit{closed}.

Applying (\textit{close}$_1$), \textit{Status}(v_{23}) is connected to a new complex non-state \( v_{23} \) with

\[
\text{Label}(v_{23}) = \text{Label}(v_7) \cup \{a : a, a : \langle A_2 \rangle \neg p\}.
\]

Notice that \( a : \langle A_2 \rangle \neg p \in \text{Reduced}(v_{23}) \). \text{Applying (\textit{form-state})}, \( v_{23} \) is connected to a new complex state \( v_{24} \) with \textit{Label}(v_{24}) = Label(v_{23}). \text{Applying (\textit{trans})}, \( v_{24} \) is connected to the existing node \( v_{10} \) with \( ELables(v_{24}, v_{10}) = \{b : \langle \sigma \rangle \langle A_2 \rangle \neg p\} \).

Observe that the assertion \( b : \langle \sigma \rangle \langle A_2 \rangle \neg p \) is not \( \lozenge \)-realizable at \( v_{24} \) w.r.t. \( v_{24} \). Hence, applying (\textit{close}$_2$) and (\textit{close}$_1$), \textit{Status}(v_{24}) is first changed to \textit{closed-wrt}((v_{24})) and then to \textit{closed}; after that, by applying (\textit{close}$_3$), the statuses of \( v_{23}, v_{21} \) and \( v_{17} \) are changed to \textit{closed} in subsequent steps.

Applying ([\textit{A}]_f), \( v_{18} \) is connected to a new complex non-state \( v_{25} \) with

\[
\text{Label}(v_{25}) = \text{Label}(v_7) \cup \{a : \langle \sigma \rangle \langle A_2 \rangle p, a : [\sigma] [\langle A_2 \rangle p]\}.
\]

Applying (\textit{\Box}_{\text{trans}}), \( v_{25} \) is connected to new complex non-states \( v_{26} \) with

\[
\text{Label}(v_{26}) = \text{Label}(v_{25}) \cup \{b : \langle A_2 \rangle p\}.
\]

Recall that \( b : \langle A_2 \rangle \neg p \) belongs to \textit{Label}(v_{18}), \textit{Reduced}(v_{18}), \textit{Reduced}(v_{18}), \textit{Reduced}(v_{25}) and \textit{Reduced}(v_{26}). \text{Applying (\textit{close})}, \textit{Status}(v_{26}) is changed to \textit{closed} (due to \( b : \langle A_2 \rangle \neg p \) and \( b : [\langle A_2 \rangle p] \)), and then, by applying (\textit{close}$_3$), the statuses of \( v_{25}, v_{18} \) and \( v_{7} \) are changed to \textit{closed} in subsequent steps.

Applying (\textit{close}$_1$), \( v_{18} \) is connected to a new complex non-state \( v_{27} \) with

\[
\text{Label}(v_{27}) = \{a : [\sigma] [\langle A_1 \rangle p] , a : p, \sigma(a, b), b : [\sigma] [\langle A_1 \rangle p], b : p, b : a, b : \langle A_2 \rangle \neg p\}.
\]

Applying (\textit{repl-nom}), \( v_{27} \) is connected to a new complex non-state \( v_{28} \) with

\[
\text{Label}(v_{28}) = \{b : [\sigma] [\langle A_1 \rangle p], b : p, \sigma(b, b), b : \langle A_3 \rangle \neg p\}.
\]

Applying ([\textit{A}]_f), \( v_{28} \) is connected to new complex non-states \( v_{29}, v_{30}, v_{31} \) with

\[
\text{Label}(v_{29}) = \{b : [\sigma] [\langle A_1 \rangle p], b : p, \sigma(b, b), b : \neg p\},
\]
\[
\text{Label}(v_{30}) = \{b : [\sigma] [\langle A_1 \rangle p], b : p, \sigma(b, b), b : \langle \sigma \rangle \langle A_3 \rangle \neg p\},
\]
\[
\text{Label}(v_{31}) = \{b : [\sigma] [\langle A_1 \rangle p], b : p, \sigma(b, b), b : \langle b? \rangle \langle A_3 \rangle \neg p\}.
\]

Applying (\textit{close}$_1$), \textit{Status}(v_{29}) is changed to \textit{closed}.
Notice that $b : [A_1] p \in Reduced(v_{30})$. Applying \((form-state)\), $v_{30}$ is connected to a new complex state $v_{32}$ with $Label(v_{32}) = Label(v_{30})$. Applying \((trans)\), $v_{32}$ is connected to a new simple non-state $v_{33}$ with

$$Label(v_{33}) = \{(A_3) \neg p, [A_1]p\}.$$

$ELabels(v_{32}, v_{33}) = \{b: (\sigma)(A_3) \neg p\}.$

Applying \(\{A\}_f\), $v_{33}$ is connected to a new simple non-state $v_{34}$ with

$$Label(v_{34}) = \{(A_3) \neg p, [\sigma][A_1]p, p\}.$$

Applying \(\{A\}_f\), $v_{34}$ is connected to the existing node $v_{12}$ and new simple non-states $v_{35}$ and $v_{36}$ with

$$Label(v_{35}) = \{(\sigma)(A_3) \neg p, [\sigma][A_1]p, p\}.$$

$$Label(v_{36}) = \{(b?)(A_3) \neg p, [\sigma][A_1]p, p\}.$$

Applying \((form-state)\), $v_{35}$ is connected to a new simple state $v_{37}$ with $Label(v_{37}) = Label(v_{35})$. Applying \((trans)\), $v_{37}$ is connected to the existing node $v_{33}$ with

$$ELabels(v_{37}, v_{33}) = \{(\sigma)(A_3) \neg p\}.$$

Applying \(\langle ? \rangle\), $v_{36}$ is connected to a new simple non-state $v_{38}$ with

$$Label(v_{38}) = \{b, (A_3) \neg p, [\sigma][A_1]p, p\}.$$

Applying \((nominal)\), $Status(v_{38})$ is changed to \textit{blocked}. Notice that $v_{38}$ is \textquotedblleft compatible\textquotedblright\ with $v_{32}$, which is the only complex state that is an ancestor of $v_{38}$, and hence, $Status(v_{32})$ is not changed (to \textit{incomplete}).

Observe that the assertion $b : (\sigma)(A_3) \neg p$ is not \(\Diamond\)-realizable at $v_{32}$ w.r.t. $v_{32}$. Hence, applying \((close_2)\) and \((close_1)\), $Status(v_{32})$ is first changed to \textit{closed-wrt}($\{v_{32}\}$) and then to \textit{closed}; after that, by applying \((close_3)\), $Status(v_{30})$ is also changed to \textit{closed}.

Applying \(\langle ? \rangle\), $v_{31}$ is connected to a new simple non-state $v_{39}$ with

$$Label(v_{39}) = \{b; [\sigma][A_1]p, b; p, \sigma(b, b), b; b, b; (A_3) \neg p\}.$$

Notice that $b : (A_3) \neg p \in Reduced(v_{39})$. Applying \((form-state)\), $v_{39}$ is connected to a new complex state $v_{40}$ with $Label(v_{40}) = Label(v_{39})$. Applying \((trans)\), $Status(v_{40})$ is changed to \textit{expanded} (without being connected to any nodes).

Observe that the assertion $b : (A_3) \neg p$ is not \(\Diamond\)-realizable at $v_{40}$ w.r.t. $v_{40}$. Hence, applying \((close_2)\) and \((close_1)\), $Status(v_{40})$ is first changed to \textit{closed-wrt}($\{v_{40}\}$) and then to \textit{closed}; after that, by applying \((close_3)\), the statuses of $v_{39}$, $v_{31}$, $v_{28}$, $v_{27}$, $v_{8}$, $v_{5} - v_{1}$, $v$ are changed to \textit{closed} in subsequent steps. Since $Status(v) = \textit{closed}$, by Theorem 1, we conclude that the given ABox $\Gamma$ is unsatisfiable.

## 5 Proofs

In this section, let $\Gamma$ be an ABox in NNF and $G = (V, E, v)$ an arbitrary $c_{HPDL}$-tableau for $\Gamma$. We first present some properties of $G$.

**Lemma 1** Let $v \in V$ and let $\mathcal{M}$ be a Kripke model. If $SType(v) = \text{simple}$, then $(Label(v))^\mathcal{M} = (FullLabel(v))^\mathcal{M}$, else $\mathcal{M} \models Label(v)$ implies $\mathcal{M} \models Reduced(v)$.

This lemma is an invariant of the tableau construction and clearly holds.
Lemma 2  Every path consisting of only non-states in G is finite.

Proof  This lemma follows from the following observations:
- If a non-state \( w \) is a successor of a non-state \( v \) then \( \text{Reduced}(w) \supset \text{Reduced}(v) \) or \( \text{FullLabel}(w) \supset \text{FullLabel}(v) \) or the number of nominals occurring in \( \text{Label}(w) \) is smaller than the number of nominals occurring in \( \text{Label}(v) \).
- For any node \( w \), \( \text{Reduced}(w) \) and \( \text{FullLabel}(w) \) are subsets of the finite set \( \text{closure}(\Gamma) \).

\[\Box\]

5.1 Complexity Analysis

We define the length of a formula (resp. program or assertion) in the base language (i.e., the language without automaton-modal operators) to be the number of occurrences of symbols in that formula (resp. program or assertion). We define the size of a set of formulas and assertions in the base language to be the sum of the lengths of its formulas and assertions.

Definition 3  The set of basic subformulas of \( \Gamma \), denoted by \( \text{bsf}(\Gamma) \), consists of all subformulas of \( \Gamma \) and their negations in NNF. The set \( \text{closure}_0(\Gamma) \) is defined to be the smallest extension of \( \text{bsf}(\Gamma) \) such that:

1. if \( [\alpha] \varphi \in \text{bsf}(\Gamma), q \in Q_{\mathcal{A}_\alpha}, \omega \) is of the form \( \sigma \) or \( \psi \) and occurs in \( \alpha \), then \([\mathcal{A}_\alpha, q] \varphi\) and \( [\omega][\mathcal{A}_\alpha, q] \varphi\) belong to \( \text{closure}_0(\Gamma) \);
2. if \( \langle \alpha \rangle \varphi \in \text{bsf}(\Gamma), q \in Q_{\mathcal{A}_\alpha}, \omega \) is of the form \( \sigma \) or \( \psi \) and occurs in \( \alpha \), then \( \langle \mathcal{A}_\alpha, q \rangle \varphi\) and \( \langle \omega \rangle \langle \mathcal{A}_\alpha, q \rangle \varphi\) belong to \( \text{closure}_0(\Gamma) \);
3. if \( \varphi \in \text{closure}_0(\Gamma) \) and \( a \) is a nominal occurring in \( \Gamma \), then \( a : \varphi \in \text{closure}_0(\Gamma) \).

The set \( \text{closure}(\Gamma) \) is defined to be the smallest extension of \( \text{closure}_0(\Gamma) \) such that, if \( \xi \in \text{closure}(\Gamma) \), both nominals \( a \) and \( b \) occur in \( \Gamma \), and \( \xi' \) is obtained from \( \xi \) by replacing every occurrence of \( b \) with \( a \), including the ones in automata of modal operators, then \( \xi' \in \text{closure}(\Gamma) \).

In what follows, if \( X \) is a set, then by \( |X| \) we denote the cardinality of \( X \).

Lemma 3  Let \( n \) be the size of \( \Gamma \). Then, \( |\text{closure}_0(\Gamma)| = O(n^4) \) and \( |\text{closure}(\Gamma)| = O(2^{f(n)}) \) for some polynomial \( f(\cdot) \).

Proof  The cardinality of \( \text{bsf}(\Gamma) \) is of rank \( O(n) \). Consider the construction of \( \text{closure}_0(\Gamma) \) by starting from \( \text{bsf}(\Gamma) \). Applying the first and second rules, we add \( O(n^2) \) formulas to \( \text{closure}_0(\Gamma) \) (note that the number of states of an automaton \( \mathcal{A}_\alpha \) is linear in the length of \( \alpha \)). After that, applying the third rule, we add \( O(n^4) \) assertions to \( \text{closure}_0(\Gamma) \). Therefore, \( |\text{closure}_0(\Gamma)| = O(n^4) \). The second assertion of the lemma clearly follows.

Let \( u \) be a complex node of \( G \). For a formula/assertion \( \xi \), by \( \text{NomRepl}(u)(\xi) \) we denote the formula/assertion obtained from \( \xi \) by replacing every nominal \( a \) with \( \text{NomRepl}(u)(a) \), including the ones in automata of modal operators. For a set \( X \) of formulas/assertions, we define

\[
\text{NomRepl}(u)(X) = \{\text{NomRepl}(u)(\xi) \mid \xi \in X\}.
\]

Lemma 4  Formulas and assertions used for the construction of any \( \text{C}_{\text{HPDL}} \)-tableau for \( \Gamma \) belong to \( \text{closure}(\Gamma) \). Furthermore, for every node \( v \) of \( G \), the cardinality of \( \text{FullLabel}(v) \) is polynomial in the size of \( \Gamma \).
5.2 Soundness

We say that the tableau calculus $C_{HPDL}$ is sound if, for any ABox $\Gamma$ in NNF and any $C_{HPDL}$-tableau $G = (V, E, v)$ for $\Gamma$, if $\Gamma$ is satisfiable, then $\text{Status}(v) \neq \text{closed}$. Conversely, $C_{HPDL}$ is complete if, for any ABox $\Gamma$ in NNF and any $C_{HPDL}$-tableau $G = (V, E, v)$ for $\Gamma$, if $\text{Status}(v) \neq \text{closed}$, then $\Gamma$ is satisfiable. These notions are similar to the ones used for proof systems for the validity problem. We refer the reader to [19] for a discussion on different notions of soundness and completeness for tableau calculi.

Our proof of soundness of the tableau calculus $C_{HPDL}$ relies on the notion of marking defined below.
**Definition 4** Let $\mathcal{M}$ be a Kripke model of $\Gamma$ and let $u$ be a complex state of $G$ such that $\text{Status}(u) \neq \text{incomplete}$ and $\mathcal{M} \models \text{FullLabel}(u)$. Let $\mathcal{O}' = \{ a \in \mathcal{O} \mid \text{NomRepl}(u)(a) = a \}$ and $V' = \{ v \in V \mid \text{SType}(v) = \text{simple} \}$. A marking of $G$ w.r.t. $\mathcal{M}$ and $u$ is a function $f : \mathcal{O}' \cup V' \to P(\Delta^{\mathcal{M}})$ with the following properties:

1. for every $a \in \mathcal{O}'$, $a^{\mathcal{M}} \in f(a)$;
2. for every $x \in \mathcal{O}' \cup V'$ and $y \in f(x)$:
   - (a) if $x \in \mathcal{O}'$, then:
     - for every $v \in V'$ such that $(u, v) \in E$, every $x : (\sigma)\varphi \in \text{ELabels}(u, v)$, and every $z \in (\text{Label}(v))^{\mathcal{M}}$ such that $(y, z) \in \sigma^{\mathcal{M}}$, we have that $z \in f(v)$;
   - (b) else if $\text{Type}(x) = \text{state}$, then:
     - for every $v \in V'$ such that $(x, v) \in E$, every $(\sigma)\varphi \in \text{ELabels}(x, v)$, and every $z \in (\text{Label}(v))^{\mathcal{M}}$ such that $(y, z) \in \sigma^{\mathcal{M}}$, we have that $z \in f(v)$;
   - (c) else if there exists $a \in \text{Label}(x)$, then $y \in f(a)$;
   - (d) else:
     - for every $v \in V'$ such that $(x, v) \in E$ and $y \in (\text{Label}(v))^{\mathcal{M}}$, we have that $y \in f(v)$.

If $f$ and $g$ are markings of $G$ w.r.t. $\mathcal{M}$ and $u$, then we say that $f$ is less than or equal to $g$, denoted by $f \subseteq g$, if $f(x) \subseteq g(x)$ for all $x \in \mathcal{O}' \cup V'$.

**Lemma 5** Let $\mathcal{M}, u, \mathcal{O}'$ and $V'$ be as in Definition 4. The smallest marking $f$ of $G$ w.r.t. $\mathcal{M}$ and $u$ exists and, if $\mathcal{M}$ is finite\(^6\), then:

1. for every $v \in V'$ and $z \in f(v)$, $z \in (\text{FullLabel}(v))^{\mathcal{M}}$;
2. for every $a \in \mathcal{O}'$, $z \in f(a)$ and $a : \varphi \in \text{FullLabel}(u)$, we have that $z \in \varphi^{\mathcal{M}}$.

**Proof** Let $\mathcal{F}$ be the set of all markings of $G$ w.r.t. $\mathcal{M}$ and $u$. It contains, among others, $\lambda x \in \mathcal{O}' \cup V'. \Delta^{\mathcal{M}}$. It is easy to see that $f = \lambda x \in \mathcal{O}' \cup V'. \bigcap \{ g(x) \mid g \in \mathcal{F} \}$ belongs to $\mathcal{F}$. Hence, $f$ is the smallest marking of $G$ w.r.t. $\mathcal{M}$ and $u$.

Now, assume that $\mathcal{M}$ is finite. We construct $f$ in another way as follows:

1. for each $a \in \mathcal{O}'$, set $f(a) := \{ a^{\mathcal{M}} \}$;
2. for each $v \in V'$, set $f(v) := \emptyset$;
3. initialize $U$ to any queue consisting of all pairs $(a, a^{\mathcal{M}})$ for $a \in \mathcal{O}'$;
4. while $U \neq \emptyset$, do:
   - (a) extract a pair $(x, y)$ from $U$;
   - (b) if $x \in \mathcal{O}'$, then:
     - for every $v \in V'$ such that $(u, v) \in E$, every $x : (\sigma)\varphi \in \text{ELabels}(u, v)$, and every $z \in (\text{Label}(v))^{\mathcal{M}}$ such that $(y, z) \in \sigma^{\mathcal{M}}$, add $z$ to $f(v)$ and $(v, z)$ to $U$;
   - (c) else if $\text{Type}(x) = \text{state}$, then:
     - for every $v \in V'$ such that $(x, v) \in E$, every $(\sigma)\varphi \in \text{ELabels}(x, v)$, and every $z \in (\text{Label}(v))^{\mathcal{M}}$ such that $(y, z) \in \sigma^{\mathcal{M}}$, add $z$ to $f(v)$ and $(v, z)$ to $U$;
   - (d) else if there exists $a \in \text{Label}(x)$, then:
     - add $y$ to $f(a)$ and $(a, y)$ to $U$;
   - (e) else:
     - for every $v \in V'$ such that $(x, v) \in E$ and $y \in (\text{Label}(v))^{\mathcal{M}}$, add $y$ to $f(v)$ and $(v, y)$ to $U$.

\(^6\) This condition can be weakened, e.g., to that $\mathcal{M}$ is finitely branching.
Since $\mathcal{M}$ is finite, the above process terminates. Observe that the construction closely reflects the definition of markings and it extends $f(x)$ for $x \in \mathcal{O}' \cup \mathcal{V}'$ minimally during the process. Hence, the resulting $f$ is the smallest marking of $G$ w.r.t. $\mathcal{M}$ and $u$. The assertions about $f$ stated in the lemma together with “if $(x, y) \in U$ then $x \in \mathcal{O}' \cup \mathcal{V}'$ and $y \in f(x)$” are invariants of the “while” loop. They hold before executing the loop because $\mathcal{M} \models FullLabel(u)$. They still hold after each execution of the step 4b, 4c or 4e because $(Label(v))^\mathcal{M} = (FullLabel(v))^\mathcal{M}$ (by Lemma 1). Consider the step 4d. We have $(x, y) \in U$ and $x \notin \mathcal{O}'$. Hence, $x \in \mathcal{V}'$, $y \in f(x)$ and $y \in (FullLabel(x))^\mathcal{M}$. Assume that $a \in Label(x)$. Thus, $y = a^\mathcal{M}$. Since $\mathcal{M} \models FullLabel(u)$, it follows that $y \in \varphi^\mathcal{M}$ for all $a : \varphi \in FullLabel(u)$. This completes the proof. \hfill $\Box$

\textbf{Remark 3} Note that our interpretation of $G$ in $\mathcal{M}$ is the smallest marking $f$ of $G$ w.r.t. $\mathcal{M}$ and $u$, where $u$ a complex state of $G$ such that $Status(u) \neq \text{incomplete}$ and $\mathcal{M} \models FullLabel(u)$. As shown below (in the proof of Lemma 6), if $\text{SType}(v) = \text{simple}$ and $f(v) \neq \emptyset$, then $Status(v) \notin \{\text{closed}, \text{closed-wrt}_{1}(u)\}$. One can define an interpretation of a tableau (see Definition 1 and recall that $G$ is a $\mathcal{C}_{\text{HPDL}}$-tableau) in $\mathcal{M}$ analogously. Then, to prove soundness of $\mathcal{C}_{\text{HPDL}}$, he or she can try to show that, if a tableau $G'$ is interpretable in $\mathcal{M}$ and $G''$ is obtained from $G'$ by applying a tableau rule, then $G''$ is interpretable in $\mathcal{M}$. Our approach is slightly different: we concentrate directly on $G$ and ignore the intermediate tableaux obtained during the construction of $G$. This approach has the advantage that $G$ is already “stable” w.r.t. the tableau rules and allows us to shorten the proof. Note that, in the case of HPDL, interpretations of tableaux in a Kripke model are quite complex, and the usual approach could result in a considerably longer proof. \hfill $\Box$

\textbf{Lemma 6} Let $\mathcal{M}$ be a finite Kripke model of $\Gamma$ and $u$ a complex state of $G$ such that $Status(u) \neq \text{incomplete}$ and $\mathcal{M} \models FullLabel(u)$. Then, $Status(u) \neq \text{closed}$. 

\textbf{Proof} Let $f : \mathcal{O}' \cup \mathcal{V}' \rightarrow P(\Delta^\mathcal{M})$ be the smallest marking of $G$ w.r.t. $\mathcal{M}$ and $u$, where $\mathcal{O}' = \{a \in \mathcal{O} | \text{NomRep}(u)(a) = a\}$ and $\mathcal{V}' = \{v \in \mathcal{V} | \text{SType}(v) = \text{simple}\}$. Let $\mathcal{V}'' = \{u\} \cup \{w \in \mathcal{V}' | f(w) \neq \emptyset\}$. We prove that, if the status of a node $v \in \mathcal{V}$ is changed to closed or closed-wrt$_{1}(u)$, then $v \notin \mathcal{V}''$, by induction on that moment.

Recall that $\mathcal{M} \models FullLabel(u)$ and, by Lemma 5, if $w \in \mathcal{V}'$ and $z \in f(w)$, then $z \in (FullLabel(w))^\mathcal{M}$. Hence, if $w \in \mathcal{V}''$, then $FullLabel(w)$ is satisfiable.

If $Status(v)$ is changed to closed by the rule (close$_1$) because there exists $o : \bot \in Label(v)$ or $a : \neg a \in Label(v)$ or $\{\xi, \overline{\xi}\} \subseteq FullLabel(v)$, then $FullLabel(v)$ is unsatisfiable and hence $v \not\in \mathcal{V}''$.

If $v \in \mathcal{V}''$ and $Status(v)$ was changed to closed by the rule (close$_1$) because $Status(v) = \text{closed-wrt}_1(v)$, then $v$ must be a complex state and thus $v = u$, and by the inductive assumption, $v \not\in \mathcal{V}''$, a contradiction.

Consider the case when $Status(v)$ is changed to closed-wrt$_1(u)$ by the rule (nominal) and, for the sake of contradiction, assume that $v \in \mathcal{V}''$. Thus, $v \in \mathcal{V}'$ and $f(v) \neq \emptyset$. Hence, $FullLabel(v)$ is satisfied at a state in $\mathcal{M}$, in particular, $\mathcal{M} \models \xi$ (where $\xi$ is the assertion mentioned in the rule (nominal)), which contradicts the facts that $\overline{\xi} \in FullLabel(u)$ and $\mathcal{M} \models FullLabel(u)$.

Observe that, if $w \in \mathcal{V}''$, $Status(w) \notin \{\text{unexpanded}, \text{closed}, \text{blocked}, \text{closed-wrt}(\ldots)\}$ and $Type(w) = \text{non-state}$, then, by Lemma 5 and Definition 4 (the property 2d), $w$ must have a successor belonging to $\mathcal{V}''$. Namely, under those assumptions about $w$, there exists $y \in f(v)$ and we have $y \in (FullLabel(w))^\mathcal{M}$, and consequently, there exists a successor $w'$ of $w$ such that $y \in (FullLabel(w'))^\mathcal{M}$, and thus $y \in f(w')$ and $w' \in \mathcal{V}''$. Therefore, if
\[ \text{Status}(v) \text{ is changed to closed or closed-wrt}_+ (u) \text{ by the rule (close}_3\text{), then, by the inductive assumption, it follows that } v \notin V'' . \]

Similarly, if \( w \in V'' \), \( \text{Status}(w) \notin \{ \text{unexpanded, closed, incomplete} \} \) and \( \text{Type}(w) = \text{state} \), then all successors of \( w \) must belong to \( V'' \). Hence, if \( \text{Status}(v) \) is changed to closed or closed-wrt\(_+\) \( (u) \) by the rule (close\(_4\)), then, by the inductive assumption, it follows that \( v \notin V'' \).

There remains the case when \( \text{Status}(v) \) is changed to closed-wrt\(_+\) \( (u) \) by the rule (close\(_2\)). For this case, it is sufficient to prove that:

1. every assertion of the form \( a: \langle A, q \rangle \varphi \) or \( a: \langle \omega \rangle \langle A, q \rangle \varphi \) in \( \text{FullLabel}(u) \) is \( \triangleleft \)-realizable at \( u \) w.r.t. \( u \),
2. if \( w \in V' \) and \( f(w) \neq \emptyset \), then every formula of the form \( \langle A, q \rangle \varphi \) or \( \langle \omega \rangle \langle A, q \rangle \varphi \) in \( \text{FullLabel}(w) \) is \( \triangleleft \)-realizable at \( w \) w.r.t. \( u \).

We prove the second assertion and omit the first one, as its proof is similar. Suppose that \( w \in V' \), \( z \in f(w) \), \( \xi \in \text{FullLabel}(w) \) and \( \xi \) is of the form \( \langle A, q \rangle \varphi \) or \( \langle \omega \rangle \langle A, q \rangle \varphi \). We need to prove that \( \xi \) is \( \triangleleft \)-realizable at \( w \) w.r.t. \( u \). By Lemma 5, \( z \in (\text{FullLabel}(w))^M \). Consider the case when \( \xi = \langle A, q \rangle \varphi \) (the other case is similar and omitted). Since \( z \in (\text{FullLabel}(w))^M \), there exist \( z_0, \ldots, z_k \in \Delta^M \) and a run \( q_0, \ldots, q_k \) of \( \langle A, q \rangle \) on a word \( \omega_1 \ldots \omega_k \) such that \( z_0 = z, z_k \in \varphi^M \) and, for each \( 1 \leq i \leq k \), if \( \omega_i \in \Sigma \) then \( (z_{i-1}, z_i) \in \omega_i^M \), else \( \omega_i = (\psi_i) \) for some \( \psi_i \) and \( z_{i-1} = z_i \) and \( z_i \in \psi_i^M \). Recall that we have \( q_0 = q, q_k \in F_A \) and \( (q_{i-1}, q_i) \in \Delta_A \) for all \( 1 \leq i \leq k \). Also note that \( z_i \in (\langle A, q_i \rangle \varphi)^M \) for all \( 0 \leq i \leq k \).

We prove that such a satisfaction of \( \xi \) at \( z \) in \( M \) is reflected by a \( \triangleleft \)-realization \( (w_0, \xi_0), \ldots, (w_h, \xi_h) \) of \( \xi \) at \( w \) w.r.t. \( u \) such that:

- for \( 0 \leq j < h \), \( w_j \in V'' \) and \( \xi_j \) is of the form \( o_j: \langle A, q_i \rangle \varphi \) (with \( 0 \leq i_j < k \)) or \( o_j: \langle \omega_i \rangle \langle A, q_i \rangle \varphi \) (with \( 0 \leq i_j < k \)),
- \( i_0 = 0, i_{h-1} = k \) and, for every \( 1 \leq j < h, i_j = i_{j-1} \) or \( i_j = i_{j-1} + 1 \).

We construct it as follows:

1. \( j := 0, w_j := w, o_j := \text{null}, i_j := 0, \xi_j := o_j: \xi; \)
2. repeat: 7\( a \) if \( \xi_j = o_j: \langle A, q_i \rangle \varphi, i_j = k \) and \( o_j: \varphi \in \text{FullLabel}(w_j) \), then \( h := j + 1, w_h := w_j, o_h := o_j \), \( \xi_h := o_h: \varphi \) and break the loop;
(b) else if \( \xi_j = o_j: \langle A, q_i \rangle \varphi, i_j < k \) and \( o_j: \langle \omega_i \rangle \langle A, q_i \rangle \varphi \in \text{FullLabel}(w_j) \), then \( j := j + 1, w_j := w_{j-1}, o_j := o_{j-1}, i_j := i_{j-1}, \xi_j := o_j: \langle \omega_i \rangle \langle A, q_i \rangle \varphi; \)
(c) else if \( \xi_j = o_j: \langle \omega_i \rangle \langle A, q_i \rangle \varphi, i_j < k \), \( \omega_i = \psi_{i+1} \) and \( o_j: \psi_{i+1}, o_j: \langle A, q_i \rangle \varphi \subseteq \text{FullLabel}(w_j) \), then \( j := j + 1, w_j := w_{j-1}, o_j := o_{j-1}, i_j := i_{j-1} + 1, \xi_j := o_j: \langle A, q_i \rangle \varphi; \)
(d) else if \( \xi_j = o_j: \langle A, q_i \rangle \varphi, i_j = k \) and \( w_j = \text{expanded by the tableau rule (} \langle A \rangle_f \rangle \) with \( \xi_j \) as the principal assertion/formula, then \( h := j + 1, o_h := o_j, \xi_h := o_h: \varphi \), let \( w_h \) be the successor of \( w_j \) such that \( \text{Label}(w_h) \) is obtained from \( \text{Label}(w_j) \) by replacing \( \xi_j \) with \( \xi_h \) and break the loop;
(e) else if \( \xi_j = o_j: \langle A, q_i \rangle \varphi, i_j < k \) and \( w_j = \text{expanded by the tableau rule (} \langle A \rangle_f \rangle \) or \( (\langle A \rangle_f) \) with \( \xi_j \) as the principal assertion/formula, then \( j := j + 1, o_j := o_{j-1}, i_j := i_{j-1}, \xi_j := o_j: \langle \omega_i \rangle \langle A, q_i \rangle \varphi \) and let \( w_j \) be the successor of \( w_{j-1} \) such that \( \xi_j \in \text{FullLabel}(w_j) \) and \( \xi_j \) is the assertion/formula obtained from \( \xi_{j-1} \) by the expansion of \( w_{j-1} \);

7 The steps 2a–2i of this loop correspond to the conditions 4a–4i of Definition 2, respectively.
(f) else if \( \xi_j = \sigma_j : \langle \omega_{i,j+1} \rangle (A, q_{i,j+1}) \varphi, i_j < k, \omega_{i,j+1} = (\psi_{i,j+1}) \) and \( w_j \) was expanded by the tableau rule \( (\varphi \gamma) \) with \( \xi_j \) as the principal assertion/formula, then: \( j := j + 1, o_j := o_{j-1}, i_j := i_{j-1} + 1, \xi_j := o_j : \langle A, q_i \rangle \varphi \) and let \( w_j \) be the unique successor of \( w_{j-1} \);

(g) else if \( w_j \) was expanded by the rule (form-state) or a static tableau rule and \( \xi_j \) is not a principal assertion/formula, then: \( j := j + 1 \), \( o_j := o_{j-1}, i_j := i_{j-1} + 1, \xi_j := \xi_{j-1} \) and let \( w_j \) be a successor of \( w_{j-1} \) such that \( z_{i,j} \in f(w_j) \);

(h) else if \( \xi_j = \sigma_j : \langle \omega_{i,j+1} \rangle (A, q_{i,j+1}) \varphi, i_j < k, \omega_{i,j+1} \in \Sigma \) and \( w_j \) was expanded by the rule (trans), then: \( j := j + 1, o_j := o_{j-1}, i_j := i_{j-1} + 1, \xi_j := o_j : \langle A, q_i \rangle \varphi \) and let \( w_j \) be the successor of \( w_{j-1} \) such that \( \xi_{j-1} \in \text{ELabels}(w_{j-1}, w_j) \);

(i) else if \( a \in \text{Label}(w_j) \), then: \( j := j + 1, w_j := u, o_j := a, i_j := i_{j-1} \) and \( \xi_j = o_j : \xi_{j-1} \).

By using properties of the marking \( f \) (Definition 4 and Lemma 5), the construction of \( G \) and the mentioned satisfaction of \( \xi \) at \( z \) in \( \mathcal{M} \), it is straightforward to check that the following properties are invariants of the “repeat” loop:

1. \( \xi_j \in \text{FullLabel}(w_j) \),
2. \( \xi_j \) is either \( o_j : \langle A, q_i \rangle \varphi \) (with \( 0 \leq i_j \leq k \)) or \( o_j : \langle \omega_{i,j+1} \rangle (A, q_{i,j+1}) \varphi \) (with \( 0 \leq i_j < k \)),
3. if \( w_j \in V' \), then \( z_{i,j} \in f(w_j) \), else \( w_j = u, o_j \in \rho' \) and \( z_{i,j} \in f(o_j) \).

By these invariants, it can be seen that the “let” instruction in the step 2g is well-defined. In particular, immediately before executing this “let” instruction: we have that \( z_{i,j-1} \in f(w_{j-1}) \) (by the inductive assumption); by Lemma 5, it follows that \( z_{i,j-1} \in (\text{FullLabel}(w_{j-1}))^\mathcal{M} \), hence, there exists a successor \( w_j \) of \( w_{j-1} \) such that \( z_{i,j-1} \in (\text{FullLabel}(w_j))^\mathcal{M} \), and by Definition 4 (of markings), \( z_{i,j-1} \in f(w_j) \), which means \( z_{i,j} \in f(w_j) \). All the other “let” instructions in the above process are also well-defined.

The mentioned process terminates because:

- During the process, if \( j \geq 1 \), then:
  - \( i_{j-1} \leq i_j \leq k \);
  - if \( i_j = i_{j-1} \), then one of the following conditions holds:
    - \( w_j \) is a successor of \( w_{j-1} \) and \( \xi_j = \xi_{j-1} \);
    - \( o_j = o_{j-1}, i_j = i_{j-1} + 1, \xi_j = o_j : \langle A, q_{i,j} \rangle \varphi \) and \( \xi_j := o_j : \langle \omega_{i,j+1} \rangle (A, q_{i,j+1}) \varphi \);
    - \( w_j = u \) and \( \xi_j = o_j : \xi_{j-1} \).

- By Lemma 2, every path consisting of only non-states in \( G \) is finite.

- In any iteration of the “repeat” loop, at least one among the steps 2a–2i is executed with effects (it either terminates the loop or increases \( j \)).

The sequence \( (w_0, \xi_0), \ldots, (w_h, \xi_h) \) is an evidence for that \( \xi \) is \( \hat{\Diamond} \)-realizable at \( w \) w.r.t. \( u \) because:

- The steps 2a–2i of the process that constructs this sequence correspond to the conditions 4a–4i of Definition 2, respectively.
- \( w_0 = w, \xi_0 = \xi, \xi_h = o_h : \varphi \) and, for \( 1 \leq j \leq h \), \( \xi_j \in \text{FullLabel}(w_j) \).
- For \( 0 \leq j \leq h \), since \( w_j \in V'' \) (either \( w_j = u \) or \( w_j \in V' \) and \( z_{i,j} \in f(w_j) \)), by the inductive assumption, \( \text{Status}(w_j) \not\in \{\text{closed}, \text{closed-wrt}_u(u)\} \).
- At the moment when considering application of the tableau rule (\( \text{close}_2 \)) to \( w \), by the inductive assumption, \( \text{Status}(u) \not\equiv \text{closed} \).

\( \square \)
Corollary 3 (Soundness) Let \( \Gamma \) be an ABox in NNF and \( G = (V,E,v) \) an arbitrary \( C_{\text{HPDL}} \)-tableau for \( \Gamma \). If \( \Gamma \) is satisfiable, then \( \text{Status}(v) \neq \emptyset \).

Proof It is known that HPDL has the finite model property [26]. Assume that \( \Gamma \) is satisfiable and let \( \mathcal{M} \) be a finite Kripke model of \( \Gamma \). There exists a path \( \pi \) in \( G \) that starts from \( v \), goes through a number of complex non-states and reaches a complex state \( u \).

If \( \mathcal{M} \models \text{FullLabel}(w) \) iff \( \mathcal{M} \models \text{FullLabel}(w') \). Since \( \mathcal{M} \models \Gamma \) and \( \text{FullLabel}(v) = \Gamma \), it follows that \( \mathcal{M} \models \text{FullLabel}(u) \). By Lemma 6, \( \text{Status}(u) \neq \emptyset \). Hence, for every node \( w \) on the path \( \pi \), starting from \( u \) and ending at \( v \), \( \text{Status}(w) \neq \emptyset \). In particular, \( \text{Status}(v) \neq \emptyset \).

5.3 Completeness

In this subsection, assume that \( \text{Status}(v) \neq \emptyset \) (where \( v \) is the root of \( G \)). Since \( \text{Status}(v) \neq \emptyset \), there exists a complex state \( u \) of \( G \) such that \( \text{Status}(u) \notin \{\emptyset, \text{incomplete}\} \). In this subsection, we fix such a \( u \). We prove that \( \Gamma \) is satisfiable by constructing a model graph \( G' \) of \( G \) w.r.t. \( u \) and then a Kripke model \( \mathcal{M} \) that corresponds to \( G' \). We start by defining model graphs (w.r.t. HPDL), which are similar to Hintikka structures.

Definition 5 A model graph is a structure \( G' = (V', E', \text{Label}', \text{ElLabels}') \), where \( E' \subseteq V' \times V' \), \( \text{ElLabels}' : E' \rightarrow P(\Sigma) \) and, for \( x \in V' \), \( \text{Label}'(x) \) is a set of formulas, such that the following properties hold for every \( x \in V' \) and every \( \varphi \in \text{Label}'(x) \):

1. \( \bot \notin \text{Label}'(x) \) and \( \overline{\varphi} \notin \text{Label}'(x) \);
2. if \( \varphi = \psi \land \chi \), then \( \{\psi, \chi\} \subseteq \text{Label}'(x) \);
3. if \( \varphi = \psi \lor \chi \), then \( \psi \in \text{Label}'(x) \) or \( \chi \in \text{Label}'(x) \);
4. if \( \varphi = \alpha \), then \( x = a \);
5. if \( \varphi = [\alpha]\psi \), \( \alpha \notin \Sigma \), \( \alpha \) is not a test and \( I_{\alpha a} = \{q_1, \ldots, q_k\} \), then \( \{[\alpha_a, q_1]\psi, \ldots, [\alpha_a, q_k]\psi \} \subseteq \text{Label}'(x) \);
6. if \( \varphi = [A,q]\psi \) and \( \delta_\varphi(q) = \{(\omega_1, q_1), \ldots, (\omega_k, q_k)\} \), then \( \{[\omega_1][A,q_1]\psi, \ldots, [\omega_k][A,q_k]\psi \} \subseteq \text{Label}'(x) \);
7. if \( \varphi = [A,q]\psi \) and \( q \in F_A \), then \( \psi \in \text{Label}'(x) \);
8. if \( \varphi = [\chi_?]\psi \), then \( \chi \in \text{Label}'(x) \) or \( \psi \in \text{Label}'(x) \);
9. if \( \varphi = [\sigma]\psi \), \( (x,y) \in E' \) and \( \sigma \in \text{ElLabels}'(x,y) \), then \( \psi \in \text{Label}'(y) \);
10. if \( \varphi = \langle \alpha \rangle \psi \), \( \alpha \notin \Sigma \), \( \alpha \) is not a test and \( I_{\alpha a} = \{q_1, \ldots, q_k\} \), then \( \{[\alpha_a, q_1]\psi, \ldots, [\alpha_a, q_k]\psi \} \cap \text{Label}'(x) \neq \emptyset \);
11. if \( \varphi = (\chi_?)\psi \), then \( \chi \notin \text{Label}'(x) \);
12. if \( \varphi = [\sigma]\psi \), then there exists \( y \) such that \( (x,y) \in E' \), \( \sigma \in \text{ElLabels}'(x,y) \), and \( \psi \in \text{Label}'(y) \);
13. if \( \varphi = [A,q]\psi \), then there exist an accepting run \( q_0, \ldots, q_k \) of the automaton \( (A,q) \) on a word \( \omega_1 \ldots \omega_k \) (with \( q_0 = q \) and \( q_k \in F_A \)) and a sequence \( x_0, \ldots, x_k \) of nodes of \( G' \) such that \( x_0 = x, \psi \in \text{Label}'(x_k) \) and, for each \( 1 \leq i \leq k \), if \( \omega_i = (\chi_?) \), then \( x_i = x_{i-1} \) and \( \{x_i,[A,q_i]\psi\} \subseteq \text{Label}'(x_i) \), else \( (x_{i-1}, x_i) \in E', \omega_i \in \text{ElLabels}'(x_{i-1}, x_i) \) and \( \{A,q_i]\psi \in \text{Label}'(x_i) \).

Definition 6 A model graph \( G' = (V', E', \text{Label}', \text{ElLabels}') \) is called a model graph of \( G \) w.r.t. \( u \) if:
Let \( v \) be a descendant of \( u \) such that it is a simple non-state and \( \text{Status}(v) \notin \{\text{closed}, \text{closed-wrt}_+(u)\} \). A saturation path of \( v \) w.r.t. \( u \) is a sequence \( v_0, v_1, \ldots, v_k \) of nodes of \( G \), with \( v_0 = v \) and \( k \geq 1 \), such that:

- \( \text{Status}(v_i) \notin \{\text{closed}, \text{closed-wrt}_+(u)\} \) for all \( 0 \leq i \leq k \),
- \( \text{Type}(v_i) = \text{non-state} \) for all \( 0 \leq i < k \) and \( \text{Type}(v_k) = \text{state} \),
- \( (v_i, v_{i+1}) \in E \) for all \( 0 \leq i < k - 1 \),
- if there exists \( a \in \text{Label}(v_{k-1}) \), then \( v_k = a \), else \( (v_{k-1}, v_k) \in E \).

By Lemma 2, each saturation path of \( v \) w.r.t. \( u \) is finite. Furthermore, if \( v_i \) is a simple non-state with \( \text{Status}(v_i) \notin \{\text{closed}, \text{closed-wrt}_+(u)\} \) and \( \text{Label}(v_i) \) does not contain any nominal, then \( v_i \) has a successor \( v_{i+1} \) with \( \text{Status}(v_{i+1}) \notin \{\text{closed}, \text{closed-wrt}_+(u)\} \). Therefore, we have the following lemma.

**Lemma 7** If \( v \) is a descendant of \( u \) such that it is a simple non-state and \( \text{Status}(v) \notin \{\text{closed}, \text{closed-wrt}_+(u)\} \), then it has at least one saturation path w.r.t. \( u \).

**Lemma 8** If \( v \) is a descendant of \( u \) with \( \text{Status}(v) \notin \{\text{closed}, \text{closed-wrt}_+(u)\} \), then every \( \xi \in \text{FullLabel}(v) \) of the form \( \langle A, q \rangle \varphi \) or \( \langle \omega \rangle \langle A, q \rangle \varphi \) has a \( \diamond \)-realization at \( v \) w.r.t. \( u \).

This lemma immediately follows from the rule \( \langle \text{close}_2 \rangle \), Definition 2 and the fact that \( \text{Status}(v) \neq \text{unexpanded} \).

**Lemma 9** There exists a model graph of \( G \) w.r.t. \( u \).

**Proof** We construct a model graph \( G' = (V', E', \text{Label}', \text{ELabels}') \) of \( G \) w.r.t. \( u \) as follows:

1. \( V' := \{a \in O \mid \text{NomRepl}(u)(a) = a\} \);
2. \( E' := \{(a, b) \mid \text{there exists } \sigma(a, b) \in \text{Label}(u)\} \);
3. for each \( a \in V' \), \( \text{Label}'(a) := \{\varphi \mid a: \varphi \in \text{FullLabel}(u)\} \cup \{a\} \);
4. for each \( (a, b) \in E' \), \( \text{ELabels}(a, b) := \{\sigma \mid \sigma(a, b) \in \text{Label}(u)\} \);
5. \( \text{realized} := \emptyset \);
6. while there exist \( x \in V' \) and \( \varphi \in \text{Label}'(x) \) such that \( \varphi \) is of the form \( \langle \sigma \rangle \psi \) and \( (x, \varphi) \notin \text{realized} \), do:
   - (a) \( x_0 := x, \sigma_0 := \sigma \);
   - (b) if \( \psi \) is of the form \( \langle A, q \rangle \psi' \), then:
     - i. if \( x \in V \), then let \( (v_0, \xi_0), \ldots, (v_{k+1}, \xi_{k+1}) \) be a \( \Diamond \)-realization of \( \varphi \) at \( x \) w.r.t. \( u \), else let \( (v_0, \xi_0), \ldots, (v_{k+1}, \xi_{k+1}) \) be a \( \Diamond \)-realization of \( x: \varphi \) at \( u \) w.r.t. \( u \);
     - ii. if \( \text{Type}(v_{k+1}) = \text{state} \), then let \( \ell = 1 \), else let \( v_{k+1}, \ldots, v_{k+\ell} \) be a saturation path of \( v_{k+1} \) w.r.t. \( u \);
iii. let $i_1, \ldots, i_h$ be all the indices such that $0 < i_1 < \ldots < i_h = k + \ell$, Type($v_{i_j}$) = state for $1 \leq j \leq h$, and for every $1 \leq j < h$, if $v_{i_j} = u$, then $\xi_{i_j}$ is of the form $a_j : \langle \sigma_j \rangle (A, q_{i_j}) \psi'$, else $\xi_{i_j}$ is of the form $\langle \sigma_j \rangle (A, q_{i_j}) \psi'$;
iv. if $v_{i_h} = u$, then let $a_h$ be a nominal such that $a_h \in Label(v_i)$, where $i$ is the greatest index such that $i < i_h$ and $v_i \neq u$;
v. for each $j$ from 1 to $h$, do:
   A. if $v_{i_j} \neq u$, then:
      - $x_j := v_{i_j}$;
      - if $x_j \notin V'$, then add $x_j$ to $V'$ and set $Label'(x_j) := FullLabel(v_{i_j})$;
   B. else $x_j := a_j$;
   C. add $(x_{j-1}, x_j)$ to $E'$ and $\sigma_{j-1}$ to $ELabels(x_{j-1}, x_j)$;

(c) else:
i. if $x \in V$, then let $v_0 = x$ and $\xi_0 = \varphi$, else let $v_0 = u$ and $\xi_0 = x : \varphi$;
ii. let $v_1$ be the successor of $v_0$ such that $\xi_0 \in ELabels(v_0, v_1)$;
iii. let $v_1, \ldots, v_i$ be a saturation path of $v_1$ w.r.t. $u$;
iv. if $v_{i_{1}} \neq u$, then:
   A. add $(x_{i_{1}-1}, x_{i_{1}})$ to $E'$ and $\sigma_{0}$ to $ELabels(x_{i_{1}-1}, x_{i_{1}})$;
   B. else:
      - let $a_1$ be a nominal such that $a_1 \in Label(v_i)$, where $i$ is the greatest index such that $i < i_1$ and $v_i \neq u$;
      - $x_1 := a_1$;
   vi. if $x_1 \notin V'$, then add $x_1$ to $V'$ and set $Label'(x_1) := FullLabel(x_1)$;
   v. else:
      - let $a_1$ be a nominal such that $a_1 \in Label(v_i)$, where $i$ is the greatest index such that $i < i_1$ and $v_i \neq u$;
      - $x_1 := a_1$;
   vi. add $(x_0, x_1)$ to $E'$ and $\sigma_0$ to $ELabels(x_0, x_1)$;

(d) add $(x, \varphi)$ to realized.

As invariants of the “while” loop in the above construction, we have that:

- if $x \in V' \cap V$, then $Status(x) \notin \{ \text{closed, closed-wrt}_+(u) \}$;
- the “let” instruction at the step 6(b)i is well-defined due to Lemma 8;
- the “let” instructions at the steps 6(b)iii and 6(b)iii are well-defined due to Lemma 7.

Also observe that:

- the “let” instruction at the step 6(b)iii is well-defined; it specifies not only $i_1, \ldots, i_h$, but also $\sigma_j$ for all $1 \leq j < h$, and $a_j$ for all $1 \leq j < h$ such that $v_{i_j} = u$;
- since $G$ is finite (by Corollary 2), the “while” loop terminates and $G'$ is finite.

Observe that the nodes added to $V'$ at the step 6 are simple states of $G$. Thus, due to the steps 1 – 4, the conditions 1 – 4 about $G'$ of Definition 6 hold. We need to prove that the conditions 1 – 13 about $G'$ of Definition 5 also hold.

The condition 1 of Definition 5 holds because $Status(u) \neq \text{closed}$ and $Status(x) \notin \{ \text{closed, closed-wrt}_+(u) \}$ for all $x \in V' \cap V$. The conditions 2, 3, 5–8, 10 and 11 of Definition 5 hold because $u$ and the nodes from $V' \cap V$ are states of $G$. The condition 4 of Definition 5 holds due to the step 3 and the fact that all the nodes added to $V'$ at the step 6 are simple states of $G$, whose statuses are different from blocked.

Consider the condition 12 of Definition 5. Let $x \in V'$, $\varphi \in Label'(x)$ and $\varphi = \langle \sigma \rangle \psi$. Consider the iteration of the “while” loop in which $(x, \varphi)$ is processed. Observe that:

- if $x \in V$, then $v_0 = x$ and $\xi_0 = \varphi$, else $v_0 = u$ and $\xi_0 = x : \varphi$,
- $v_0$ is a state and was expanded by the rule (trans),
- $v_1$ is the successor of $v_0$ in $G$ such that $\xi_0 \in ELabels(v_0, v_1)$,
- $\psi \in FullLabel(v_1)$.
\[ \text{FullLabel}(v_1) \subseteq \text{Label}(x_1). \]

Thus, we can take \( y = x_1 \) to satisfy the condition 12 of Definition 5.

Consider the condition 9 of Definition 5. Suppose that \( x'' \in V', \psi'' \in \text{Label}'(x'') \), \( \sigma'' = [\sigma'']|\psi'' \), \( (x'', y'') \in E' \) and \( \sigma'' \in \text{ELabels}'(x'', y'') \). We need to prove that \( \psi'' \in \text{Label}'(y'') \).

There are the following two cases:

- Case \( \{x'', y''\} \subseteq O \): We must have that \( \sigma''(x'', y'') \in \text{Label}(u) \) and \( x'' : [\sigma'']|\psi'' \in \text{FullLabel}(u) \). Since \( u \) is a state, it follows that \( y'' : \psi'' \in \text{FullLabel}(u) \), and hence \( \psi'' \in \text{Label}'(y'') \).

- Case \( (x'', y'') \) was added to \( E' \) at the step 6: This happened in some iteration of the “while” loop with \( x'' = x_{j-1} \), \( y'' = x_j \) and \( \sigma'' = \sigma_{j-1} \), for some \( 1 \leq j \leq h \) in the case of the step 6(b)vC, or for \( j = 1 \) in the case of the step 6(b)vI. Let \( i_0 = 0 \). Observe that:

  - if \( x_{j-1} \in V \), then \( v_{i_{j-1}} = x_{j-1} \) and \( \xi_{i_{j-1}} \) is of the form \( \langle \sigma_{j-1} \rangle \chi \), else \( v_{i_{j-1}} = u \) and \( \xi_{i_{j-1}} \) is of the form \( x_{j-1} : \langle \sigma_{j-1} \rangle \chi \).
  - \( v_{i_{j-1}} \) is a state and was expanded by the rule (trans).
  - \( v_{i_{j-1}+1} \) is the successor of \( v_{i_{j-1}} \) in \( G \) with \( \xi_{i_{j-1}} \in \text{ELabels}(v_{i_{j-1}}, v_{i_{j-1}+1}) \).
  - \( \psi'' \in \text{FullLabel}(v_{i_{j-1}+1}) \) since \( [\sigma'']|\psi'' \in \text{Label}'(x_{j-1}) \) and \( \sigma'' = \sigma_{j-1} \).
  - if \( v_j \neq u \), then \( x_j = v_j \), else \( v_j = a_j \).
  - \( \text{FullLabel}(v_{i_{j-1}+1}) \subseteq \text{Label}'(x_j) \).

Hence, \( \psi'' \in \text{Label}'(x_j) \), which means \( \psi'' \in \text{Label}'(y'') \).

Consider the condition 13 of Definition 5. Suppose that \( x \in V' \), \( \varphi' \in \text{Label}'(x) \) and \( \varphi = \langle A, q' \rangle \psi' \). We need to show that there exist an accepting run \( p_0, \ldots, p_n \) of the automaton \( (A, q') \) on a word \( \omega_1 \ldots \omega_n \) with \( p_0 = q' \) and \( p_n \in F_A \) and a sequence \( \gamma_0, \ldots, \gamma_n \) of nodes of \( G' \) such that:

\[
\begin{align*}
\gamma_0 &= x, \psi' \in \text{Label}'(\gamma_0) \quad \text{and, for each} \quad 1 \leq i \leq n, \quad \text{if} \ \omega_i = (x_i), \\
\gamma_i &= \gamma_{i-1} \quad \text{and} \quad \{\chi_i, \langle A, p_i \rangle \psi'\} \subseteq \text{Label}'(\gamma_i), \quad \text{else} \quad \gamma_{i-1} \quad \text{and} \quad \gamma_i \subseteq E'.
\end{align*}
\]

If \( x \in V \), then let \( \bar{v} = x \) and \( \xi = \varphi' \), else let \( v = u \), \( \xi = x : \varphi' \) and \( a_0 = x \).

Since \( \text{Status}(u) \neq \text{closed} \) and by the mentioned invariants of the “while” loop, \( \text{Status}(v) \notin \{\text{closed}, \text{closed-wrt}_u(u)\} \). By Lemma 8, it follows that \( \xi \) has a \( \hat{\cdot} \)-realization at \( v \) w.r.t. \( u \). The case when that \( \hat{\cdot} \)-realization is of the form \( (v, \xi_0), (v, \xi_1), \ldots, (v, \xi_k) \) is easy and omitted. Consider the other case when that \( \hat{\cdot} \)-realization has a prefix \( (v, \xi_{-m}), \ldots, (v, \xi_0) \) with \( \xi_0 = \langle \sigma \rangle \langle A, q \rangle \psi' \) when \( x \in V \), or \( \xi_0 = x : \langle \sigma \rangle \langle A, q \rangle \psi' \) when \( x \in O \). Let \( \varphi = \langle \sigma \rangle \langle A, q \rangle \psi' \).

Consider the moment after the iteration of the “while” loop that processes the pair \( (x, \varphi) \).

We will refer to the values of the variables used in the loop at that moment. We have \( v_0 = v \), \( x_0 = x \) and \( \sigma_0 = \sigma \). Let \( q_0 = q_i, i_0 = 0 \) and \( i_{-1} = -m \). For each \( 0 \leq j \leq h \), let \( i_{j,1}, \ldots, i_{j,s_j} \) be a sequence of indices with a maximum length and a minimum value of \( i_{j,1} \), such that:

- \( i_{j-1} < i_{j,1} < i_{j,2} < \ldots < i_{j,s_j} < i_j \),
- if \( j = h \), then \( i_{j,s_j} < k \),
- for any \( i \) among \( i_{j,1}, \ldots, i_{j,s_j} \), \( \xi_i \) is of the form \( o_i : (x_i) \psi_i q_i \),
- for any \( 1 \leq t < s_j \), with \( i = i_{j,t} \) and \( i' = i_{j,t+1} \), we have that \( q_i \neq q_i \).

\[ \text{If } x_j \in V, \text{then } o_j = \text{null}, \text{else either } o_j = \text{null} \text{ or } o_j = a_j. \]

\[ \text{We require this due to the condition 4g of Definition 2. It may exclude transitions of the form } (q_i, x_i, q_i) \text{ of the automaton } A \text{ but does not cause loss of generality when proving the property (2) at a later step.} \]
We construct the mentioned accepting run \( p_0, \ldots, p_n \) of \((A, q')\), the mentioned word \( \omega_1 \ldots \omega_n \) and the mentioned sequence \( y_0, \ldots, y_n \) of nodes of \( G' \) as follows:

- \( p_0 := q', y_0 := x, n := 0 \);
- for each \( j \) from 0 to \( h \), do:
  - for each \( t \) from 1 to \( s_j \), do:
    - \( n := n + 1 \) and let \( i \) denote \( i_{j,t} \);
    - \( \chi_n := \chi_i, \omega_n := (\chi_n ?), p_n := q_i, y_n := x_j \);
  - if \( j < h \), then:
    - \( n := n + 1 \) and let \( i \) denote \( i_j \);
    - \( \omega_n := \sigma_j, p_n := q_i, y_n := x_{j+1} \).

It is straightforward to check that the property (2) holds.

\[ \square \]

**Definition 8** Let \( G' = (V', E', Label', ELLabels') \) be a model graph of \( G \) w.r.t. \( u \). A Kripke model \( \mathcal{M} \) corresponds to \( G' \) w.r.t. \( u \) if:

- \( \Delta^\mathcal{M} = V' \),
- \( p^\mathcal{M} = \{ x \in V' \mid p \in \text{Label}'(x) \} \) for \( p \in \text{PROP} \),
- \( \sigma^\mathcal{M} = \{ (x, y) \in E' \mid \sigma \in \text{ELLabels}'(x, y) \} \) for \( \sigma \in \Sigma \),
- \( a^\mathcal{M} = \text{NomRepl}(u)(a) \) for \( a \in \mathcal{O} \) with \( \text{NomRepl}(u)(a) \) specified.

\[ \square \]

Clearly, there exist Kripke models corresponding to \( G' \) w.r.t. \( u \). They differ from each other only in interpreting nominals \( a \) with \( \text{NomRepl}(u)(a) \) unspecified.

**Lemma 10** Let \( G' = (V', E', Label', ELLabels') \) be a model graph of \( G \) w.r.t. \( u \) and \( \mathcal{M} \) a Kripke model corresponding to \( G' \) w.r.t. \( u \). Then:

1. for every \( x \in V' \) and every \( \varphi \in \text{Label}'(x) \), we have \( x \in \varphi^\mathcal{M} \),
2. \( \mathcal{M} \models \text{FullLabel}(u) \),
3. \( \mathcal{M} \models \text{Label}(v) \).

**Proof** We define \( \preceq \) to be the smallest transitive binary relation on the set of formulas in NNF of the extended language such that:

- if \( \psi \) is syntactically a subformula of \( \chi \), then \( \psi \preceq \chi \), (e.g., \( \psi \preceq \psi \), \( \psi \preceq (\psi ?)\chi \), \( \chi \preceq (\psi ?)\chi \), \( \psi \preceq [A, q]\psi \), etc.),
- if \( \alpha \notin \Sigma \), \( \alpha \) is not a test and \( q \in I_{\bar{\alpha}} \), then \( \langle \bar{\alpha}, q \rangle \psi \preceq \langle \alpha \rangle \psi \) and \([\bar{\alpha}, q] \psi \preceq [\alpha] \psi \),
- if \( (\psi ?) \in \Sigma_A \), then \( \psi \preceq \langle A, q \rangle \chi \) and \( \overline{\psi} \preceq [A, q] \chi \),
- \( \overline{\overline{\psi}} \preceq [\psi ?] \chi \).

Observe that \( \preceq \) is a well-founded order. We prove the first assertion of the lemma by induction on \( \preceq \). Let \( x \in V' \) and \( \varphi \in \text{Label}'(x) \).

- The base cases occur when \( \varphi \) is of the form \( p \) or \( a \). The hypothesis for the case \( \varphi = p \) follows directly from Definition 8. Consider the case when \( \varphi = a \). By the condition 4 of Definition 5, \( x = a \). By the condition 1 of Definition 6, it follows that \( \text{NomRepl}(u)(a) = a \). Consequently, by Definition 8, we can derive that \( x \in a^\mathcal{M} \).\(^{10}\)

\(^{10}\) Recall the remark in the caption of Fig. 1 on page 5.
Consider the case when \( \varphi = \langle A, q \rangle \psi \). By the condition 13 of Definition 5, there exist an accepting run \( q_0, \ldots, q_k \) of the automaton \((A, q)\) on a word \( \omega_1 \ldots \omega_k \) (with \( q_0 = q \) and \( q_k \in F_A \)) and a sequence \( x_0, \ldots, x_k \) of nodes of \( G' \) such that \( x_0 = x, \psi \in Label'(x_k) \) and, for each \( 1 \leq i \leq k \), if \( \omega_i = (\chi_i ?) \), then \( x_i = x_{i-1} \) and \( \{\chi_i, (A, q_i) \psi\} \subseteq Label'(x_i) \), else \( \langle x_{i-1}, x_i \rangle \in E' \), \( \omega_i \in ELabels'(x_{i-1}, x_i) \) and \( (A, q_i) \psi \in Label(x_i) \). Thus, by the inductive assumption, \( x_k \in \psi^M \) and, for \( 1 \leq i \leq k \), if \( \omega_i = (\chi_i ?) \), then \( x_i \in \chi_i^M \).

Consequently, it follows that \( (x_0, x_k) \in (\omega_1; \ldots; \omega_k)^M \), and hence \( x_0 \in (\langle A, q \rangle \psi)^M \), which means \( x \in \psi^M \).

Consider the case when \( \varphi = [A, q] \psi \). Let \((x, y) \in ([A, q])^M \). We need to prove that \( y \in \psi^M \). Since \((x, y) \in ([A, q])^M \), there exist an accepting run \( q_0, \ldots, q_k \) of the automaton \((A, q)\) on a word \( \omega_1 \ldots \omega_k \) (with \( q_0 = q \) and \( q_k \in F_A \)) and a sequence \( x_0, \ldots, x_k \) of nodes of \( G' \) such that \( x_0 = x, x_k = y \) and, for each \( 1 \leq i \leq k \), if \( \omega_i = (\chi_i ?) \), then \( x_i = x_{i-1} \) and \( x_i \in \chi_i^M \), else \( \langle x_{i-1}, x_i \rangle \in E' \) and \( \omega_i \in ELabels'(x_{i-1}, x_i) \). Thus, by the inductive assumption, for \( 1 \leq i \leq k \), if \( \omega_i = (\chi_i ?) \), then \( \vec{\chi_i} \notin Label'(x_i) \).

By the conditions 6, 8 and 9 of Definition 5, it follows that, for each \( i \) from 1 to \( k \), \([A, q_k] \psi \in Label'(x_i) \). Thus, \([A, q_k] \psi \in Label'(y) \). Since \( q_k \in F_A \), by the condition 7 of Definition 5, it follows that \( \psi \in Label'(y) \). By the inductive assumption, this implies \( y \in \psi^M \).

The other cases for the induction step can be dealt with in a straightforward way by using the conditions of Definition 5.

The second assertion of the lemma follows from the first one due to Definitions 6 and 8. The third assertion follows from the second one. Namely, there exists a path \( v_0, \ldots, v_k \) in \( G \) such that \( v_0 = v \) and \( v_k = u \), and by the applied tableau rules, for every \( i \) from \( k - 1 \) down to 0, \( M \models FullLabel(v_i) \) follows from \( M \models FullLabel(v_{i+1}) \).

**Corollary 4** (Completeness) Let \( \Gamma \) be an ABox in NNF and \( G = (V, E, v) \) an arbitrary \( C_{HPDL} \)-tableau for \( \Gamma \). If \( \text{Status}(v) \neq \text{closed} \), then \( \Gamma \) is satisfiable.

This corollary immediately follows from the third assertion of Lemma 10 (since \( Label(v) = \Gamma \)).

### 6 Concluding Remarks

We have given the first direct tableau procedure with the \( \text{ExpTime} \) complexity for deciding HPDL and proved that it is sound and complete. The procedure uses global caching, a technique that not only guarantees the \( \text{ExpTime} \) complexity, but probably also increases efficiency.\(^{11}\) As HPDL can be used as a description logic for representing and reasoning about terminological knowledge, our procedure is useful for practical applications.

Although the techniques for dealing with nominals were known for tableaux with global caching for the description logic \( SHIQ \) \(^{32} \) and tableaux with global state caching \(^{22} \) for the description logic \( SHIQ \) \(^{29} \), combining them with checking fulfillment of eventualities for HPDL is not a trivial task. This is reflected in our proofs of soundness and completeness of the tableau calculus \( C_{HPDL} \).

In our decision procedure, any expansion strategy can be used for constructing a tableau. One can give the rule \((\text{close}_1)\) the highest priority and give unary static rules a higher priority.

---

\(^{11}\) There are evidences supporting the claim that global caching increases efficiency for reasoning in the DLs \( \text{ACC} \) \(^{20}, \text{SHIQ} \) \(^{30}, \text{Section 7}\). We do not have evidences for the case of HPDL. In general, efficiency strongly depends also on optimizations and the applied search strategy.
than for non-unary static rules. One may choose the depth-first expansion strategy, globally cache only simple nodes, keep complex nodes only for the current path of complex nodes, and is still guaranteed to have the ExpTime complexity for the algorithm. Checking fulfillment of eventualities can be done on-the-fly as in [21] or periodically for the whole graph or at special moments for subgraphs (for example, when the subgraph rooted at a node has been “fully expanded” and no ♦-realization goes out from that subgraph).

Our decision procedure has been designed to simplify the presentation and leaves space for improvement. It has been implemented for the TGC2 system [30] with various optimizations. For example, a sequence of expansions by unary static rules is done in one step to eliminate intermediate non-states with only one successor, automata in modal operators are minimized, and different control strategies (for expanding the constructed tableau) are mixed. Further optimization techniques like propagation of unsatisfiability cores or compacting ABoxes by using bisimilarity are planned to be implemented for the next version of TGC2. We refer the reader to [30] for more details about this system.

Our tableau method can be extended for Graded HPDL using the techniques from [31] and for Converse-HPDL using the techniques from [22,28,29].

Acknowledgements We would like to thank the anonymous reviewers for many comments and suggestions, which have helped us to improve the paper significantly.

References

1. Areces, C., ten Cate, B.: Hybrid logics. In: Blackburn, P., Benthein, J.V., Wolter, F. (eds.) Handbook of Modal Logic, Studies in Logic and Practical Reasoning, vol. 3, pp. 821–868. Elsevier, New York (2007)
2. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica 69, 5–40 (2001)
3. Blackburn, P.: Internalizing labelled deduction. J. Log. Comput. 10(1), 137–168 (2000)
4. Blackburn, P., ten Cate, B.: Pure extensions, proof rules, and hybrid axiomatics. Studia Logica 84(2), 277–322 (2006)
5. Bolander, T., Blackburn, P.: Termination for hybrid tableaus. J. Log. Comput. 17(3), 517–554 (2007)
6. Bolander, T., Blackburn, P.: Terminating tableau calculi for hybrid logics extending K. Electr. Notes Theor. Comput. Sci. 231, 21–39 (2009)
7. Bolander, T., Braüner, T.: Tableau-based decision procedures for hybrid logic. J. Log. Comput. 16(6), 737–763 (2006)
8. Braüner, T.: Why does the proof-theory of hybrid logic work so well? J. Appl. Non-Class. Log. 17(4), 521–543 (2007)
9. Braüner, T.: Hybrid logic. In: Zalta, E. (ed.) The Stanford Encyclopedia of Philosophy, summer, 2017th edn. Stanford University, Metaphysics Research Lab (2017)
10. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D.: Reasoning in expressive description logics. In: Robinson, J.A., Voronkov, A. (eds) Handbook of Automated Reasoning, vol. 2, pp. 1581–1634. Elsevier and MIT Press (2001)
11. Cerrito, S., Mayer, M.C.: Nominal substitution at work with the global and converse modalities. In: Beklemishev, L.D., Goranko, V., Shehtman, V.B. (eds) Advances in Modal Logic 8, Papers from the Eighth Conference on Advances in Modal Logic held in Moscow, Russia, 24–27 August 2010, pp. 59–76. College Publications (2010)
12. Divroodi, A., Nguyen, L.: On bisimulations for description logics. Inf. Sci. 295, 465–493 (2015)
13. Donini, F., Massacci, F.: ExpTime tableaux for ALCC. Art. Intell. 124, 87–138 (2000)
14. Dunin-Keplicz, B., Nguyen, L., Szalas, A.: Converse-PDL with regular inclusion axioms: a framework for MAS logics. J. Appl. Non-Class. Log. 21(1), 61–91 (2011)
15. Fischer, M., Ladner, R.: Propositional dynamic logic of regular programs. J. Comput. Syst. Sci. 18(2), 194–211 (1979)
16. Giacomo, G.D., Lenzerini, M.: TBox and ABox reasoning in expressive description logics. In: Proceedings of KR’1996, pp. 316–327 (1996)
17. Goré, R.: And-or tableaux for fixpoint logics with converse: LTL, CTL, PDL and CPDL. In: Proceedings of IJCAR 2014, LNCS, vol. 8562, pp. 26–45. Springer (2014)
18. Goré, R., Nguyen, L.: Analytic cut-free tableaux for regular modal logics of agent beliefs. In: Proceedings of CLIMA VIII, LNCS, vol. 5056, pp. 274–289. Springer (2007)
19. Goré, R., Nguyen, L.: ExpTime tableaux for $\mathcal{ALC}$ using sound global caching. J. Autom. Reason. 50(4), 355–381 (2013)
20. Goré, R., Postniece, L.: An experimental evaluation of global caching for $\mathcal{ALC}$ (system description). In: Proceedings of IJCAR 2008, LNCS, vol. 5195, pp. 299–305. Springer (2008)
21. Goré, R., Widmann, F.: An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In: R. Schmidt (ed.) Proceedings of CADE-22, LNAI 5663, pp. 437–452. Springer (2009)
22. Goré, R., Widmann, F.: Optimal and cut-free tableaux for propositional dynamic logic with converse. In: Proceedings of IJCAR 2010, LNCS, vol. 6173, pp. 225–239. Springer (2010)
23. Hansen, J.: Terminating tableaux for dynamic epistemic logics. Electr. Notes Theor. Comput. Sci. 262, 141–156 (2010)
24. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, New York (2000)
25. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. J. Autom. Reason. 39(3), 249–276 (2007)
26. Kaminski, M., Smolka, G.: A goal-directed decision procedure for Hybrid PDL. J. Autom. Reason. 52(4), 407–450 (2014)
27. Kritsimallis, A.: Tableaux with partial caching for Hybrid PDL with satisfaction statements. In: Proceedings of ICTAC 2017, LNCS, vol. 10580, pp. 229–247. Springer (2017)
28. Nguyen, L.: Cut-free ExpTime tableaux for Converse-PDL extended with regular inclusion axioms. In: Proc. of KES-AMSTA’2013, Frontiers in Artificial Intelligence and Applications, vol. 252, pp. 235–244. IOS Press (2013)
29. Nguyen, L.: ExpTime tableaux with global state caching for the description logic $\mathcal{SHIO}$. Neurocomputing 146, 249–263 (2014)
30. Nguyen, L.: Design of the tableau reasoner TGC2 for description logics. Int. J. Softw. Eng. Knowl. Eng. 26(8), 1315–1333 (2016)
31. Nguyen, L.: ExpTime tableaux with global caching for graded propositional dynamic logic. Fundam. Inf. 147(2–3), 261–288 (2016)
32. Nguyen, L., Golińska-Pilarek, J.: An ExpTime tableau method for dealing with nominals and qualified number restrictions in deciding the description logic $\mathcal{SHOIQ}$. Fundam. Inf. 135(4), 433–449 (2014)
33. Nguyen, L., Szalas, A.: Checking consistency of an ABox w.r.t. global assumptions in PDL. Fundam. Inf. 102(1), 97–113 (2010)
34. Pratt, V.: A near-optimal method for reasoning about action. J. Comput. Syst. Sci. 20(2), 231–254 (1980)
35. Sattler, U., Vardi, M.: The hybrid $\mu$-calculus. In: Proceedings of IJCAR 2001, LNCS, vol. 2083, pp. 76–91. Springer (2001)
36. Schild, K.: A correspondence theory for terminological logics: preliminary report. In: Proceedings of IJCAI’1991, pp. 466–471 (1991)
37. Seligman, J.: Internalization: The case of hybrid logics. J. Log. Comput. 11(5), 671–689 (2001)
38. Tzakova, M.: Tableau calculi for hybrid logics. In: Proceedings of TABLEAUX’99, LNCS, vol. 1617, pp. 278–292. Springer (1999)