# ExpTime Tableaux with Global Caching for Graded Propositional Dynamic Logic

**Linh Anh Nguyen**

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

*Division of Knowledge and System Engineering for ICT*
*Faculty of Information Technology*
*Ton Duc Thang University*
*Ho Chi Minh City, Vietnam*

**Abstract.** We present the first direct tableau decision procedure for graded PDL, which uses global caching and has ExpTime (optimal) complexity when numbers are encoded in unary. It shows how to combine checking fulfillment of existential star modalities with integer linear feasibility checking for tableaux with global caching. As graded PDL can be used as a description logic for representing and reasoning about terminological knowledge, our procedure is useful for practical applications.

## 1. Introduction

Propositional dynamic logic (PDL) is a well-known modal logic [1, 2]. Originally, it was developed as a logic for reasoning about programs. However, its extensions are also used for other purposes. For example, converse-PDL with regular inclusion axioms (CPDL$_{reg}$) can be used as a framework for multiagent logics [3]. As a variant of PDL, $\mathcal{ALC}_{reg}$ is a description logic for representing and reasoning about terminological knowledge. Several extensions of $\mathcal{ALC}_{reg}$ have been studied by the description logic community [4].

The satisfiability problem in PDL is EXPTIME-complete [1]. In [5], Pratt gave a tableau decision procedure with global caching for deciding PDL. In [6], Nguyen and Szałas reformulated that procedure and extended it for dealing with checking consistency of an ABox w.r.t. a TBox in PDL (where PDL is

treated as a description logic). The work [7] by Abate et al. gives another tableau decision procedure with global caching for PDL, which propagates unfulfillment of existential star modalities on-the-fly. There are also tableau decision procedures with global caching or state global caching for CPDL (PDL with converse) [8, 9] and CPDL$_{reg}$ [3].

Graded modal logics allow graded modalities for reasoning about the number of successor states with a certain property. They have attracted attention from many researchers.[1] In description logic, the counterpart of graded modalities is qualified number restrictions. Some well-known EXPTIME description logics with qualified number restrictions are $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$. The description logic corresponding to graded CPDL is $\mathcal{CIQ}$ [10]. De Giacomo and Lenzerini [10] proved that the satisfiability problem in $\mathcal{CIQ}$ is EXPTIME-complete when numbers are encoded in unary. Tobies [11] proved that the satisfiability problem in $\mathcal{SHIQ}$ is EXPTIME-complete even when numbers are encoded in binary.

This paper is a revised and extended version of our workshop paper [12].[2] In this paper, we present the first direct tableau decision procedure for GPDL (graded PDL). Our procedure uses global caching and has EXPTIME (optimal) complexity when numbers are encoded in unary. It shows how to combine checking fulfillment of existential star modalities with integer linear feasibility checking [13] for tableaux with global caching.

As GPDL can be used as a description logic for representing and reasoning about terminological knowledge, our procedure is useful for practical applications. The existing reasoners for description logics like RACER, HermiT, FaCT++, Pellet currently cannot deal with PDL and its extensions, partially because dealing with fulfillment of existential star modalities and its interaction with qualified number restrictions is a complicated problem and no efficient methods for that problem were known before. Our method presented in this paper solves that problem and can be further extended to deal with other features of description logics.

As related work on automated reasoning in GPDL and its extensions, De Giacomo and Lenzerini gave methods for translating the satisfiability problem in $\mathcal{CIQ}$ into $\mathcal{CIF}$ (a variant of CPDL with functionalities) [10], and in $\mathcal{CIF}$ into CPDL [14]. This established the complexity EXPTIME-complete for $\mathcal{CIQ}$ (and GPDL) when numbers are encoded in unary. However, this indirect method cannot give an efficient decision procedure for GPDL because it is not scalable w.r.t. numbers in graded modalities (i.e., qualified number restrictions). The translation from $\mathcal{CIQ}$ to $\mathcal{CIF}$ [10] may result in a formula with a quadratic length, and similarly for the translation from $\mathcal{CIF}$ to CPDL [14]. A further discussion on this will be given in the conclusions.

The rest of this paper is structured as follows. In Section 2, we present the notation and semantics of GPDL and recall automaton-modal operators [2, 3], which are used for our tableaux. We omit the feature of "global assumptions" as they can be expressed in PDL (by "local assumptions"). In Section 3, we present our tableau calculus for GPDL, starting with the data structure, the tableau rules and ending with the corresponding tableau decision procedure and its properties. In Section 4, we give examples for illustrating our procedure. Conclusions are given in Section 5.

[1] See `http://www.cs.man.ac.uk/~ezolin/ml/` for a list of related publications.

[2] The tableau calculus given in [12] is sound, but not complete. It has been corrected for the current paper and all of the results have been proved.

# 2. Preliminaries

## 2.1. Graded 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). We denote elements of $\Sigma$ by letters like $\sigma$ and $\varrho$, and elements of $\mathcal{PROP}$ by letters like $p$ and $q$.

A *Kripke model* is a pair $\mathcal{M} = (\Delta^{\mathcal{M}}, \cdot^{\mathcal{M}})$, where $\Delta^{\mathcal{M}}$ is a set of *states* and $\cdot^{\mathcal{M}}$ is an interpretation function that maps each proposition $p \in \mathcal{PROP}$ to a subset $p^{\mathcal{M}}$ of $\Delta^{\mathcal{M}}$ and each atomic program $\sigma \in \Sigma$ to a binary relation $\sigma^{\mathcal{M}}$ on $\Delta^{\mathcal{M}}$. Intuitively, $p^{\mathcal{M}}$ is the set of states in which $p$ is true and $\sigma^{\mathcal{M}}$ is the binary relation consisting of pairs (input_state, output_state) of the program $\sigma$. A Kripke model $\mathcal{M}$ is said to be *finitely-branching* if, for every $x \in \Delta^{\mathcal{M}}$, the number of $y \in \Delta^{\mathcal{M}}$ such that $(x, y) \in \sigma^{\mathcal{M}}$ for some $\sigma \in \Sigma$ is finite.

*Formulas* and *programs* of the *base language* of GPDL are defined respectively by the following grammar, where $p \in \mathcal{PROP}$, $\sigma \in \Sigma$ and $n$ is a natural number:

$$\varphi \quad ::= \quad \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \langle\alpha\rangle\varphi \mid [\alpha]\varphi \mid \geq n\, \sigma.\varphi \mid \leq n\, \sigma.\varphi$$

$$\alpha \quad ::= \quad \sigma \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?$$

Notice that we use the notation $\geq n\, \sigma.\varphi$ and $\leq n\, \sigma.\varphi$ as in description logic instead of $\langle\sigma\rangle_{\geq n}\varphi$ and $\langle\sigma\rangle_{\leq n}\varphi$ (as the latter do not look "dual" to each other).

We use letters like $\alpha$, $\beta$ to denote programs, and $\varphi$, $\psi$, $\xi$ 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 set-theoretical union of $\alpha$ and $\beta$,
- $\alpha^*$ stands for the reflexive and transitive closure of $\alpha$,
- $\varphi?$ stands for the test operator.

Informally, a formula $\langle\alpha\rangle\varphi$ 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 $\geq n\, \sigma.\varphi$ (resp. $\leq n\, \sigma.\varphi$) represents the set of states $x$ such that the program $\sigma$ has transitions from $x$ to at least (resp. at most) $n$ pairwise different states satisfying $\varphi$.

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

We write $\mathcal{M}, w \models \varphi$ to denote that $w \in \varphi^{\mathcal{M}}$. For a set $\Gamma$ of formulas, we denote $\Gamma^{\mathcal{M}} = \bigcap\{\varphi^{\mathcal{M}} \mid \varphi \in \Gamma\}$ and write $\mathcal{M}, w \models \Gamma$ to denote that $w \in \Gamma^{\mathcal{M}}$. If $\mathcal{M}, w \models \varphi$ (resp. $\mathcal{M}, w \models \Gamma$), then we say that $\mathcal{M}$ *satisfies* $\varphi$ (resp. $\Gamma$) *at* $w$, and that $\varphi$ (resp. $\Gamma$) is *satisfied at* $w$ in $\mathcal{M}$.

A formula is in the *negation normal form* (NNF) if it does not use $\rightarrow$ and it uses $\neg$ only immediately before propositions, and furthermore, it does not contain subformulas of the form $\geq 0\, \sigma.\varphi$ or $\leq 0\, \sigma.\varphi$. Every formula can be transformed to an equivalent formula in NNF. By $\overline{\varphi}$ we denote the NNF of $\neg\varphi$.

$$
\begin{array}{rclcrcl}
(\alpha;\beta)^{\mathcal{M}} & = & \alpha^{\mathcal{M}} \circ \beta^{\mathcal{M}} & & (\alpha \cup \beta)^{\mathcal{M}} & = & \alpha^{\mathcal{M}} \cup \beta^{\mathcal{M}} \\
(\alpha^*)^{\mathcal{M}} & = & (\alpha^{\mathcal{M}})^* & & (\varphi?)^{\mathcal{M}} & = & \{(x,x) \mid x \in \varphi^{\mathcal{M}}\} \\
\top^{\mathcal{M}} & = & \Delta^{\mathcal{M}} & & \bot^{\mathcal{M}} & = & \emptyset \\
(\neg\varphi)^{\mathcal{M}} & = & \Delta^{\mathcal{M}} \setminus \varphi^{\mathcal{M}} & & (\varphi \to \psi)^{\mathcal{M}} & = & (\neg\varphi \vee \psi)^{\mathcal{M}} \\
(\varphi \wedge \psi)^{\mathcal{M}} & = & \varphi^{\mathcal{M}} \cap \psi^{\mathcal{M}} & & (\varphi \vee \psi)^{\mathcal{M}} & = & \varphi^{\mathcal{M}} \cup \psi^{\mathcal{M}} \\
(\langle\alpha\rangle\varphi)^{\mathcal{M}} & = & \{x \in \Delta^{\mathcal{M}} \mid \exists y((x,y) \in \alpha^{\mathcal{M}} \wedge y \in \varphi^{\mathcal{M}})\} & & & & \\
([\alpha]\varphi)^{\mathcal{M}} & = & \{x \in \Delta^{\mathcal{M}} \mid \forall y((x,y) \in \alpha^{\mathcal{M}} \to y \in \varphi^{\mathcal{M}})\} & & & & \\
(\geq n\, \sigma.\varphi)^{\mathcal{M}} & = & \{x \in \Delta^{\mathcal{M}} \mid \#\{y \in \Delta^{\mathcal{M}} \mid (x,y) \in \sigma^{\mathcal{M}} \wedge y \in \varphi^{\mathcal{M}}\} \geq n\} & & & & \\
(\leq n\, \sigma.\varphi)^{\mathcal{M}} & = & \{x \in \Delta^{\mathcal{M}} \mid \#\{y \in \Delta^{\mathcal{M}} \mid (x,y) \in \sigma^{\mathcal{M}} \wedge y \in \varphi^{\mathcal{M}}\} \leq n\} & & & &
\end{array}
$$

Figure 1. Interpretation of complex programs and complex formulas.

## 2.2. Automaton-Modal Operators

The *alphabet* $\Sigma(\alpha)$ of a program $\alpha$ and the *regular language* $\mathcal{L}(\alpha)$ generated by $\alpha$ are specified as follows:[3]

$$
\begin{array}{ll}
\Sigma(\sigma) = \{\sigma\} & \mathcal{L}(\sigma) = \{\sigma\} \\
\Sigma(\varphi?) = \{\varphi?\} & \mathcal{L}(\varphi?) = \{\varphi?\} \\
\Sigma(\beta;\gamma) = \Sigma(\beta) \cup \Sigma(\gamma) & \mathcal{L}(\beta;\gamma) = \mathcal{L}(\beta).\mathcal{L}(\gamma) \\
\Sigma(\beta \cup \gamma) = \Sigma(\beta) \cup \Sigma(\gamma) & \mathcal{L}(\beta \cup \gamma) = \mathcal{L}(\beta) \cup \mathcal{L}(\gamma) \\
\Sigma(\beta^*) = \Sigma(\beta) & \mathcal{L}(\beta^*) = (\mathcal{L}(\beta))^*
\end{array}
$$

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

We will use letters like $\omega$ to denote either an atomic program from $\Sigma$ or a test (of the form $\varphi?$). A word $\omega_1 \ldots \omega_k \in \mathcal{L}(\alpha)$ can be treated as the program $(\omega_1; \ldots; \omega_k)$, especially when interpreted in a Kripke model.

Recall that a *finite automaton* $A$ over an alphabet $\Sigma(\alpha)$ is a tuple $\langle \Sigma(\alpha), Q, I, \delta, F \rangle$, where $Q$ is a finite set of states, $I \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma(\alpha) \times Q$ 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 $\mathcal{L}(A)$.

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 $\delta_A(q) = \{(\omega, q') \mid (q, \omega, q') \in \delta_A\}$.

[3]Note that $\Sigma(\alpha)$ contains not only atomic programs but also expressions of the form $(\varphi?)$, and a program $\alpha$ is a regular expression over its alphabet $\Sigma(\alpha)$.

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 $\mathcal{M}$ as follows:

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

For each program $\alpha$, let $\mathbb{A}_\alpha$ be a finite automaton recognizing the regular language $\mathcal{L}(\alpha)$. The automaton $\mathbb{A}_\alpha$ 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 $\mathbb{A}_\alpha$ 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*, 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 $(A, q)$ as a program with semantics specified by (1).

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

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

The condition $(x, x_k) \in (\omega_1; \dots; \omega_k)^{\mathcal{M}}$ means there exist states $x_0 = x, x_1, \dots, x_{k-1}$ of $\mathcal{M}$ such that, for each $1 \leq i \leq k$, if $\omega_i \in \Sigma$ then $(x_{i-1}, x_i) \in \omega_i^{\mathcal{M}}$, else $\omega_i = (\psi_i?)$ for some $\psi_i$ and $x_{i-1} = x_i$ and $x_i \in \psi_i^{\mathcal{M}}$. 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 GPDL

In this section, we present a tableau calculus for checking whether a given finite set of formulas in NNF is satisfiable. We specify the data structure, the tableau rules, the corresponding tableau decision procedure and state its properties.

## 3.1. The Data Structure

A *tableau* is a rooted graph $G = (V, E, \nu)$, where $V$ is a set of nodes, $E \subseteq V \times V$ is a set of edges, $\nu \in V$ is the root, each node $v \in V$ has a number of attributes, and each edge $(v, w)$ may be labeled by a set $ELabels(v, w) \subseteq \Sigma$. The attributes of a tableau node $v$ are:

- $Type(v) \in \{\mathsf{state}, \mathsf{non\text{-}state}\}$,
- $Status(v) \in \{\mathsf{unexpanded}, \mathsf{expanded}, \mathsf{closed}\}$,
- $Label(v)$, which is a finite set of formulas, called the *label* of $v$,
- $RFmls(v)$, which is a finite set of so called *reduced formulas* of $v$,
- $ILConstraints_0(v)$, which is a finite set of integer linear constraints.

We call $v$ a *state* if $Type(v) = \mathsf{state}$, and a *non-state* otherwise. An edge outgoing from a node $v$ is labeled iff $Type(v) = \mathsf{state}$. $RFmls(v) \neq \emptyset$ only when $Type(v) = \mathsf{non\text{-}state}$. If $(v, w) \in E$, then we call $v$ a *predecessor* of $w$ and $w$ a *successor* of $v$.

$ILConstraints_0(v)$ is available only when $Type(v) = \mathsf{state}$ and $Status(v) \neq \mathsf{unexpanded}$. The constraints use variables $x_{w,\sigma}$ indexed by a pair $(w, \sigma)$ such that $(v, w) \in E$ and $\sigma \in ELabels(v, w)$. Such a variable specifies how many copies of the successor $w$ will be created for $v$ using the edge label $\sigma$.

We apply global caching in the sense that, if $v_1$ and $v_2$ are different nodes, then either $Type(v_1) \neq Type(v_2)$ or $Label(v_1) \neq Label(v_2)$ or $RFmls(v_1) \neq RFmls(v_2)$.

We define

$$\begin{aligned} FullLabel(v) &= Label(v) \cup RFmls(v), \\ ILConstraints(v) &= ILConstraints_0(v) \ \cup \\ &\quad \{x_{w,\sigma} = 0 \mid (v, w) \in E,\ \sigma \in ELabels(v, w),\ Status(w) = \mathsf{closed}\}. \end{aligned}$$

## 3.2. Tableau Rules

Our tableau calculus $\mathcal{C}_{\mathrm{GPDL}}$ for the GPDL is specified by a number of static rules, the rule $(\mathit{form\text{-}state})$, the transitional rule and the rule $(\mathit{close})$. The last rule is used to change the status of a node to closed, while the other rules are used to expand a node. Static rules are written downwards, with a set of formulas above the line as the *premise*, which represents the label of the node to which the rule is applied, and a number of sets of formulas below the line as the *(possible) conclusions*, which represent the labels of the successor nodes resulting from the application of the rule. Possible conclusions of 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 $\Gamma$, $X$, $Y$ to denote sets of formulas and write $\Gamma, \varphi$ to denote $\Gamma \cup \{\varphi\}$. The static rules of $\mathcal{C}_{\mathrm{GPDL}}$ are specified in Table 1. For each of them, the distinguished formula of the premise is called the *principal formula* of the rule. A static rule $(\rho)$ is *applicable* to a node $v$ if the following conditions hold:

- $Status(v) = \mathsf{unexpanded}$ and $Type(v) = \mathsf{non\text{-}state}$,
- the premise of the rule is equal to $Label(v)$,
- the conditions accompanied with $(\rho)$ are satisfied,
- the principal formula of $(\rho)$ does not belong to $RFmls(v)$.

The last condition means that, if the principal formula belongs to $RFmls(v)$, then $(\rho)$ has been applied to an ancestor node of $v$ that corresponds to the same state in the intended Kripke model as $v$, and therefore should not be applied again.

If $(\rho)$ is a static rule applicable to $v$, then the application is as follows:

- Let $\varphi$ be the principal formula and $X_1, \ldots, X_k$ the possible conclusions of $(\rho)$.
- For each $1 \leq i \leq k$, do $\mathtt{ConToSucc}(v, \mathsf{non\text{-}state}, X_i, RFmls(v) \cup \{\varphi\}, \mathsf{null})$, which is specified on page 8.
- $Status(v) := \mathsf{expanded}$.

The tableau rule $(\mathit{form\text{-}state})$ is applicable to a node $v$ if $Status(v) = \mathsf{unexpanded}$, $Type(v) = \mathsf{non\text{-}state}$ and no static rule is applicable to $v$. Application of this rule to such a node $v$ is done by calling $\mathtt{ConToSucc}(v, \mathsf{state}, Label(v), \emptyset, \mathsf{null})$ and setting $Status(v) := \mathsf{expanded}$.

| |
|---|
| $(\wedge)\ \dfrac{X, \varphi \wedge \psi}{X, \varphi, \psi}$ $\qquad$ $(\vee)\ \dfrac{X, \varphi \vee \psi}{X, \varphi \mid X, \psi}$ |
| if $\alpha \notin \Sigma,\ \alpha$ is not a test, and $I_{\mathbb{A}_\alpha} = \{q_1, \ldots, q_k\}$ : <br> $(aut_\Box)\ \dfrac{X, [\alpha]\varphi}{X, [\mathbb{A}_\alpha, q_1]\varphi, \ldots, [\mathbb{A}_\alpha, q_k]\varphi}$ <br> $(aut_\Diamond)\ \dfrac{X, \langle\alpha\rangle\varphi}{X, \langle\mathbb{A}_\alpha, q_1\rangle\varphi \mid \ldots \mid X, \langle\mathbb{A}_\alpha, q_k\rangle\varphi}$ |
| if $\delta_A(q) = \{(\omega_1, q_1), \ldots, (\omega_k, q_k)\}$ and $q \notin F_A$ : <br> $([\mathrm{A}])\ \dfrac{X, [A, q]\varphi}{X, [\omega_1][A, q_1]\varphi, \ldots, [\omega_k][A, q_k]\varphi}$ <br> $(\langle\mathrm{A}\rangle)\ \dfrac{X, \langle A, q\rangle\varphi}{X, \langle\omega_1\rangle\langle A, q_1\rangle\varphi \mid \ldots \mid X, \langle\omega_k\rangle\langle A, q_k\rangle\varphi}$ |
| if $\delta_A(q) = \{(\omega_1, q_1), \ldots, (\omega_k, q_k)\}$ and $q \in F_A$ : <br> $([\mathrm{A}]_f)\ \dfrac{X, [A, q]\varphi}{X, [\omega_1][A, q_1]\varphi, \ldots, [\omega_k][A, q_k]\varphi, \varphi}$ <br> $(\langle\mathrm{A}\rangle_f)\ \dfrac{X, \langle A, q\rangle\varphi}{X, \langle\omega_1\rangle\langle A, q_1\rangle\varphi \mid \ldots \mid X, \langle\omega_k\rangle\langle A, q_k\rangle\varphi \mid X, \varphi}$ |
| $(\Box_?)\ \dfrac{X, [\psi?]\varphi}{X, \overline{\psi} \mid X, \varphi}$ $\qquad$ $(\Diamond_?)\ \dfrac{X, \langle\psi?\rangle\varphi}{X, \psi, \varphi}$ |
| if $X$ does not contain any $\geq n\, \sigma.\varphi$ with $n \geq 1$ : <br> $(\Diamond_{\geq})\ \dfrac{X, \langle\sigma\rangle\varphi}{X, \langle\sigma\rangle\varphi, \geq 1\, \sigma.\varphi}$ |

Table 1. The static rules of $\mathcal{C}_{\mathrm{GPDL}}$.

The transitional rule $(trans)$ (on page 9) is applicable to a node $v$ if $Status(v) = \mathsf{unexpanded}$ and $Type(v) = \mathsf{state}$. We give here an explanation for the tableau rule $(trans)$. To satisfy a requirement $(\geq n\, \sigma.\varphi) \in Label(v)$, one can first create a successor $w$ of $v$ specified by the pair $(\sigma, X)$ computed at

**Function** NewSucc($v, type, label, rFmls, eLabel$)

**Global data**: a rooted graph $(V, E, \nu)$.
**Purpose**: create a new successor for $v$.

create a new node $w$, $V := V \cup \{w\}$, $E := E \cup \{(v, w)\}$;
$Type(w) := type$, $Status(w) :=$ unexpanded;
$Label(w) := label$, $RFmls(w) := rFmls$;
**if** $Type(v) =$ state **then** $ELabels(v, w) := \{eLabel\}$;
**return** $w$;

**Function** ConToSucc($v, type, label, rFmls, eLabel$)

**Global data**: a rooted graph $(V, E, \nu)$.
**Purpose**: connect a node $v$ to a successor, which is created if necessary.

**if** *there exists a node* $w \in V$ *such that* $Type(w) = type$, $Label(w) = label$ *and* $RFmls(w) = rFmls$ **then**
    $E := E \cup \{(v, w)\}$;
    **if** $Type(v) =$ state **then** $ELabels(v, w) := ELabels(v, w) \cup \{eLabel\}$;
**else**
    $w :=$ `NewSucc`$(v, type, label, rFmls, eLabel)$;
**return** $w$;

the step 2, where $X$ presents $Label(w)$, and then clone $w$ to create $n$ successors for $v$ (or only record the intention somehow). The label of $w$ contains only formulas necessary for realizing the requirement $\geq n\, \sigma.\varphi$ and the related ones of the form $[\sigma]\psi$ at $v$. To satisfy requirements of the form $\leq n'\, \sigma.\varphi'$ at $v$, we tend to use only copies of such nodes like $w$ extended with either $\varphi'$ or $\overline{\varphi'}$ (for easy counting) as well as the mergers of such extended nodes. So, we first start with the set $\mathcal{E}$ constructed at the step 2, which consists of pairs with information about successors to be created for $v$. We then modify $\mathcal{E}$ by taking into account necessary extensions for the nodes (see the step 4). After that we continue modifying $\mathcal{E}$ by taking into account also appropriate mergers of nodes (see the step 9). Successors for $v$ are created at the step 13. The number of copies of a node $w$ that are intended to be used as successors of $v$ with an edge label $\sigma$ is represented by the variable $x_{w,\sigma}$ (we will not actually create such copies). The set $ILConstraints_0(v)$ consisting of appropriate constraints about such variables are set at the steps 14 and 15.

**Definition 3.1.** Let $\varphi \in FullLabel(v)$ be of the form $\langle A, q\rangle\psi$ or $\langle\omega\rangle\langle A, q\rangle\psi$. We say that $\varphi$ is $\Diamond$-*realizable at* $v$ if either $Status(v) =$ unexpanded, or $Status(v) =$ expanded and some of the following conditions hold:

1. $v$ is expanded by the tableau rule $(\langle \mathrm{A}\rangle_f)$, $\varphi = \langle A, q\rangle\psi$ is the principal formula and the successor $w$ of $v$ whose label is obtained from $Label(v)$ by replacing $\varphi$ by $\psi$ has $Status(w) \in \{$unexpanded, expanded$\}$;
2. $v$ is expanded by the tableau rule $(\langle \mathrm{A}\rangle)$ or $(\langle \mathrm{A}\rangle_f)$, $\varphi = \langle A, q\rangle\psi$ is the principal formula, $w$ is a successor of $v$ and the formula $\varphi' \in Label(w)$ obtained from $\varphi$ is $\Diamond$-realizable at $w$;
3. $v$ is expanded by the tableau rule $(\Diamond_?)$, $\varphi = \langle \xi? \rangle\langle A, q\rangle\psi$ is the principal formula and $\langle A, q\rangle\psi$ is $\Diamond$-realizable at the unique successor of $v$;

**The tableau rule** trans($v$)

$\mathcal{E} := \emptyset$, $\mathcal{E}' := \emptyset$;
**foreach** $(\geq n\, \sigma.\varphi) \in \mathit{Label}(v)$ **do**
  $\mathcal{E} := \mathcal{E} \cup \{(\sigma, X)\}$, where $X = \{\varphi\} \cup \{\psi \mid [\sigma]\psi \in \mathit{Label}(v)\}$
**foreach** $(\leq n\, \sigma.\varphi) \in \mathit{Label}(v)$ **do**
  **foreach** $(\sigma, X) \in \mathcal{E}$ **do**
    **if** $\{\varphi, \overline{\varphi}\} \cap X \neq \emptyset$ **then** $\mathcal{E}' := \mathcal{E}' \cup \{(\sigma, X)\}$
    **else** $\mathcal{E}' := \mathcal{E}' \cup \{(\sigma, X \cup \{\varphi\}), (\sigma, X \cup \{\overline{\varphi}\})\}$
  $\mathcal{E} := \mathcal{E}'$, $\mathcal{E}' := \emptyset$;
**repeat**
  **foreach** $(\leq n\, \sigma.\varphi) \in \mathit{Label}(v)$, $(\sigma, X) \in \mathcal{E}$ *and* $(\sigma, X') \in \mathcal{E}$ *such that* $\varphi \in X \cap X'$, $(\sigma, X \cup X') \notin \mathcal{E}$ *and* $X \cup X'$ *does not contain any pair of the form* $\psi$, $\overline{\psi}$ **do**
    add $(\sigma, X \cup X')$ to $\mathcal{E}$ (i.e., the merger of $(\sigma, X)$ and $(\sigma, X')$ is added to $\mathcal{E}$)
**until** *no tuples were added to* $\mathcal{E}$ *during the last iteration*;
**foreach** $(\sigma, X) \in \mathcal{E}$ **do** ConToSucc($v$, non-state, $X, \emptyset, \sigma$);
$\mathit{ILConstraints}_0(v) := \{x_{w,\sigma} \geq 0 \mid (v, w) \in E \text{ and } \sigma \in \mathit{ELabels}(v, w)\}$ ;
**foreach** $\varphi \in \mathit{Label}(v)$ **do**
  **if** $\varphi$ *is of the form* $\geq n\, \sigma.\psi$ **then** add to $\mathit{ILConstraints}_0(v)$ the constraint $\sum\{x_{w,\sigma} \mid (v, w) \in E, \sigma \in \mathit{ELabels}(v, w), \psi \in \mathit{Label}(w)\} \geq n$;
  **if** $\varphi$ *is of the form* $\leq n\, \sigma.\psi$ **then** add to $\mathit{ILConstraints}_0(v)$ the constraint $\sum\{x_{w,\sigma} \mid (v, w) \in E, \sigma \in \mathit{ELabels}(v, w), \psi \in \mathit{Label}(w)\} \leq n$;
$\mathit{Status}(v) :=$ expanded;

4. $v$ is expanded by a static tableau rule, $\varphi$ is not the principal formula and some of the following conditions hold:
   (a) $\varphi = \langle A, q\rangle\psi$, $q \in F_A$ and $\psi \in \mathit{FullLabel}(v)$,
   (b) $\varphi = \langle A, q\rangle\psi$, $(q, \omega, q') \in \delta_A$, $\langle\omega\rangle\langle A, q'\rangle\psi$ belongs to $\mathit{FullLabel}(v)$ and is $\Diamond$-realizable at $v$,
   (c) $\varphi = \langle\xi?\rangle\langle A, q\rangle\psi$, $\{\xi, \langle A, q\rangle\psi\} \subseteq \mathit{FullLabel}(v)$ and $\langle A, q\rangle\psi$ is $\Diamond$-realizable at $v$,
   (d) $\varphi$ is $\Diamond$-realizable at a successor of $v$;
5. $v$ is expanded by the tableau rule $(\mathit{form\text{-}state})$ and $\varphi$ is $\Diamond$-realizable at the unique successor of $v$;
6. $v$ is expanded by the tableau rule $(\mathit{trans})$, $\varphi = \langle\sigma\rangle\langle A, q\rangle\psi$, $w$ is a successor of $v$, $\sigma \in \mathit{ELabels}(v, w)$, $\langle A, q\rangle\psi \in \mathit{Label}(w)$, $\mathit{ILConstraints}(v) \cup \{x_{w,\sigma} \geq 1\}$ is feasible and $\langle A, q\rangle\psi$ is $\Diamond$-realizable at $w$. □

Observe that the notion of $\Diamond$-realizability is defined inductively and the cases 1 and 4a are base cases (note that for the case 1 we also have $q \in F_A$).

The tableau rule $(\mathit{close})$ is specified as follows: set $\mathit{Status}(v) :=$ closed if $\mathit{Status}(v) \neq$ closed and some of the following conditions hold:

1. $\bot \in \mathit{Label}(v)$ or there exists $\{\varphi, \overline{\varphi}\} \subseteq \mathit{FullLabel}(v)$;

2. $Type(v) = \mathsf{non\text{-}state}$, $Status(v) = \mathsf{expanded}$ and, for every $(v, w) \in E$, $Status(w) = \mathsf{closed}$;
3. $Type(v) = \mathsf{state}$, $Status(v) = \mathsf{expanded}$ and $ILConstraints(v)$ is infeasible;
4. some $\varphi \in FullLabel(v)$ of the form $\langle A, q\rangle\psi$ or $\langle\omega\rangle\langle A, q\rangle\psi$ is not $\Diamond$-realizable at $v$.

### 3.3. Checking Unsatisfiability

Let $\Gamma$ be a finite set of formulas in NNF. A $\mathcal{C}_{\mathrm{GPDL}}$-*tableau* for $\Gamma$ is a tableau $G = (V, E, \nu)$ constructed as follows. At the beginning, $V = \{\nu\}$, $E = \emptyset$ and the attributes of the root $\nu$ are specified as follows: $Type(\nu) = \mathsf{non\text{-}state}$, $Status(\nu) = \mathsf{unexpanded}$, $Label(\nu) = \Gamma$ and $RFmls(\nu) = \emptyset$. Then, while $Status(\nu) \neq \mathsf{closed}$ and there is some tableau rule $(\rho)$ applicable to some node $v$, choose such a pair $((\rho), v)$ and apply $(\rho)$ to $v$.[4] Observe that the set of all formulas that may appear in the contents of the nodes of $G$ is finite (see Lemma A.2). Due to global caching, $G$ is finite and can be effectively constructed (see Lemma A.5). The following theorem immediately follows from Corollaries B.9 and B.20, which are given and proved in the appendix.

**Theorem 3.2. (Soundness and Completeness)**
Let $\Gamma$ be a finite set of formulas in NNF and $G = (V, E, \nu)$ an arbitrary $\mathcal{C}_{\mathrm{GPDL}}$-tableau for $\Gamma$. Then, $\Gamma$ is unsatisfiable iff $Status(\nu) = \mathsf{closed}$. □

To check satisfiability of a finite set $\Gamma$ of formulas in NNF, one can construct a $\mathcal{C}_{\mathrm{GPDL}}$-tableau $G = (V, E, \nu)$ for $\Gamma$ and return "no" when $Status(\nu) = \mathsf{closed}$, or "yes" otherwise. We call this the $\mathcal{C}_{\mathrm{GPDL}}$-*tableau decision procedure*. Various optimization techniques [15] can be applied to this procedure.

**Corollary 3.3.** The $\mathcal{C}_{\mathrm{GPDL}}$-tableau decision procedure has EXPTIME complexity when numbers are encoded in unary. □

This corollary immediately follows from Lemma A.5, which is given and proved in the appendix.

## 4. Illustrative Examples

**Example 4.1.** Consider

$$\Gamma = \{\langle\sigma^*\rangle p,\ \neg p,\ [\sigma;\sigma;\sigma^*]\neg p,\ [\sigma](\neg p \vee \neg q),\ \geq 1000\, \sigma.q,\ \leq 1000\, \sigma.(p \vee q)\},$$

which is already in NNF, and let

$$\begin{array}{lcll} A_1 & = & \mathbb{A}_{\sigma^*} & = (\{\sigma\}, \{0\}, \{0\}, \{(0, \sigma, 0)\}, \{0\}), \\ A_2 & = & \mathbb{A}_{\sigma;\sigma;\sigma^*} & = (\{\sigma\}, \{0, 1, 2\}, \{0\}, \{(0, \sigma, 1), (1, \sigma, 2), (2, \sigma, 2)\}, \{2\}). \end{array}$$

A $\mathcal{C}_{\mathrm{GPDL}}$-tableau $G = (V, E, \nu)$ for $\Gamma$ is constructed as follows:

1. At the beginning, $G$ contains only the root $\nu$ with $Type(\nu) = \mathsf{non\text{-}state}$ and $Label(\nu) = \Gamma$.

[4] 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 status closed.

2. Applying $(aut_\Diamond)$ to $\nu$, this node is connected to a new non-state $v_1$ with

$$Label(v_1) = \Gamma - \{\langle\sigma^*\rangle p\} \cup \{\langle A_1, 0\rangle p\}.$$

3. Applying $(aut_\Box)$ to $v_1$, this node is connected to a new non-state $v_2$ with

$$Label(v_2) = Label(v_1) - \{[\sigma;\sigma;\sigma^*]\neg p\} \cup \{[A_2, 0]\neg p\}.$$

4. Applying $([\mathrm{A}])$ to $v_2$, this node is connected to a new non-state $v_3$ with

$$Label(v_3) = Label(v_2) - \{[A_2, 0]\neg p\} \cup \{[\sigma][A_2, 1]\neg p\}.$$

5. Applying $(\langle\mathrm{A}\rangle_f)$ to $v_3$ using the principal formula $\langle A_1, 0\rangle p$, this node is connected to two new non-states $v_4$ and $v_5$ with

$$\begin{array}{rcl} Label(v_4) & = & Label(v_3) - \{\langle A_1, 0\rangle p\} \cup \{p\} \\ Label(v_5) & = & Label(v_3) - \{\langle A_1, 0\rangle p\} \cup \{\langle\sigma\rangle\langle A_1, 0\rangle p\}. \end{array}$$

6. Since $\{p, \neg p\} \subseteq Label(v_4)$, applying the tableau rule $(close)$ to $v_4$, this node gets status closed.

7. Applying $(\Diamond_{\geq})$ to $v_5$, this node is connected to a new non-state $v_6$ with

$$\begin{array}{rcl} Label(v_6) & = & Label(v_5) \cup \{\geq 1\, \sigma.\langle A_1, 0\rangle p\} \\ & = & \{\langle\sigma\rangle\langle A_1, 0\rangle p,\ \geq 1\, \sigma.\langle A_1, 0\rangle p,\ \neg p,\ [\sigma][A_2, 1]\neg p,\ [\sigma](\neg p \vee \neg q), \\ & & \geq 1000\, \sigma.q,\ \leq 1000\, \sigma.(p \vee q)\}. \end{array}$$

8. Applying $(form\text{-}state)$ to $v_6$, we connect it to a new state $v_7$ with $Label(v_7) = Label(v_6)$.

9. Applying $(trans)$ to $v_7$, this state is connected to new non-states $v_8$–$v_{13}$, with $ELabels(v_7, v_i) = \{\sigma\}$ for $8 \leq i \leq 13$, and

$$\begin{array}{rcl} Label(v_8) & = & \{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ \neg p \vee \neg q,\ p \vee q\} \\ Label(v_9) & = & \{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ \neg p \vee \neg q,\ \neg p \wedge \neg q\} \\ Label(v_{10}) & = & \{q,\ [A_2, 1]\neg p,\ \neg p \vee \neg q,\ p \vee q\} \\ Label(v_{11}) & = & \{q,\ [A_2, 1]\neg p,\ \neg p \vee \neg q,\ \neg p \wedge \neg q\} \\ Label(v_{12}) & = & Label(v_8) \cup Label(v_{10}) \\ Label(v_{13}) & = & Label(v_9) \cup Label(v_{11}). \end{array}$$

$ILConstraints_0(v_7)$ consists of $x_{v_i,\sigma} \geq 0$, for $8 \leq i \leq 13$, and the following:

$$\begin{array}{rcl} x_{v_8,\sigma} + x_{v_9,\sigma} + x_{v_{12},\sigma} + x_{v_{13},\sigma} & \geq & 1 \\ x_{v_{10},\sigma} + x_{v_{11},\sigma} + x_{v_{12},\sigma} + x_{v_{13},\sigma} & \geq & 1000 \\ x_{v_8,\sigma} + x_{v_{10},\sigma} + x_{v_{12},\sigma} & \leq & 1000. \end{array}$$

10. Consider the node $v_9$.

    - Applying $(\wedge)$ to $v_9$ using the principal formula $\neg p \wedge \neg q$, this node is connected to a new non-state $v_{14}$ with $Label(v_{14}) = \{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ \neg p \vee \neg q,\ \neg p,\ \neg q\}$.
    - Applying $(\vee)$ to $v_{14}$ using the principal formula $\neg p \vee \neg q$, this node is connected (twice) to a new non-state $v_{15}$ with $Label(v_{15}) = \{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ \neg p,\ \neg q\}$.
    - Applying $([\mathrm{A}])$ to $v_{15}$ using the principal formula $[A_2, 1]\neg p$, this node is connected to a new non-state $v_{16}$ with $Label(v_{16}) = \{\langle A_1, 0\rangle p,\ [\sigma][A_2, 2]\neg p,\ \neg p,\ \neg q\}$.
    - Applying $(\langle \mathrm{A}\rangle_f)$ to $v_{16}$ using the principal formula $\langle A_1, 0\rangle p$, this node is connected to two new non-states $v_{17}$ and $v_{18}$ with

    $$\begin{aligned} Label(v_{17}) &= \{p,\ [\sigma][A_2, 2]\neg p,\ \neg p,\ \neg q\}, \\ Label(v_{18}) &= \{\langle\sigma\rangle\langle A_1, 0\rangle p,\ [\sigma][A_2, 2]\neg p,\ \neg p,\ \neg q\}. \end{aligned}$$

    - Since $\{p, \neg p\} \subseteq Label(v_{17})$, the tableau rule $(close)$ changes the status of $v_{17}$ to closed.
    - Applying $(\Diamond_{\geq})$ to $v_{18}$, this node is connected to a new non-state $v_{19}$ with $Label(v_{19}) = \{\langle\sigma\rangle\langle A_1, 0\rangle p,\ \geq 1\, \sigma.\langle A_1, 0\rangle p,\ [\sigma][A_2, 2]\neg p,\ \neg p,\ \neg q\}$.
    - Applying $(form\text{-}state)$ to $v_{19}$, this node is connected to a new state $v_{20}$ with $Label(v_{20}) = Label(v_{19})$.
    - Applying $(trans)$ to $v_{20}$, this node is connected to a new non-state $v_{21}$ with $Label(v_{21}) = \{\langle A_1, 0\rangle p,\ [A_2, 2]\neg p\}$.
    - Applying $([\mathrm{A}]_f)$ to $v_{21}$, this node is connected to a new non-state $v_{22}$ with $Label(v_{22}) = \{\langle A_1, 0\rangle p,\ \neg p,\ [\sigma][A_2, 2]\neg p\}$.
    - Applying $(\langle \mathrm{A}\rangle_f)$ to $v_{22}$, this node is connected to two new non-states $v_{23}$ and $v_{24}$ with

    $$\begin{aligned} Label(v_{23}) &= \{p,\ \neg p,\ [\sigma][A_2, 2]\neg p\}, \\ Label(v_{24}) &= \{\langle\sigma\rangle\langle A_1, 0\rangle p,\ \neg p,\ [\sigma][A_2, 2]\neg p\}\}. \end{aligned}$$

    - Since $\{p, \neg p\} \subseteq Label(v_{23})$, the tableau rule $(close)$ changes the status of $v_{23}$ to closed.
    - Applying $(\Diamond_{\geq})$ to $v_{24}$, this node is connected to a new non-state $v_{25}$ with $Label(v_{25}) = \{\langle\sigma\rangle\langle A_1, 0\rangle p,\ \geq 1\, \sigma.\langle A_1, 0\rangle p,\ \neg p,\ [\sigma][A_2, 2]\neg p\}\}$.
    - Applying $(form\text{-}state)$ to $v_{25}$, this node is connected to a new state $v_{26}$ with $Label(v_{26}) = Label(v_{25})$.
    - Applying $(trans)$ to $v_{26}$, this node is connected to the existing non-state $v_{21}$.
    - Observe that the formula $\langle A_1, 0\rangle p$ is not $\Diamond$-realizable at $v_{21}$. The tableau rule $(close)$ changes the status of $v_{21}$ to closed. By applying this tableau rule in the propagation manner, the statuses of the nodes $v_{20}$, $v_{19}$, $v_{18}$, $v_{16}$, $v_{15}$, $v_{14}$, $v_9$ are changed to closed one after the other.

11. Consider the node $v_{12}$. We have $\{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ q,\ \neg p \vee \neg q\} \subseteq Label(v_{12})$. Similarly as for $v_9$, it can be seen that there will not be any $\Diamond$-realization for $\langle A_1, 0\rangle p$ at $v_{12}$. As a consequence, $Status(v_{12})$ will be changed at some step to closed by the tableau rule $(close)$.

12. Observe that $\{q, \neg p \wedge \neg q\} \subset \mathit{Label}(v_{11}) \subset \mathit{Label}(v_{13})$. It is easy to see that $\mathit{Status}(v_{11})$ and $\mathit{Status}(v_{13})$ will be changed at some steps to closed.

13. Consider the moment when the statuses of the nodes $v_9$, $v_{11}$, $v_{12}$ and $v_{13}$ have been changed to closed. The set $\mathit{ILConstraints}(v_7)$ (which extends $\mathit{ILConstraints}_0(v_7)$ with $x_{v_i,e} = 0$ for $i \in \{9, 11, 12, 13\}$) is reduced to the following one w.r.t. feasibility:

$$\begin{array}{rcl} x_{v_8,\sigma} & \geq & 1 \\ x_{v_{10},\sigma} & \geq & 1000 \\ x_{v_8,\sigma} + x_{v_{10},\sigma} & \leq & 1000. \end{array}$$

Clearly, it is infeasible. As a consequence, $\mathit{Status}(v_7)$ is changed to closed by the tableau rule $(\mathit{close})$. By applying this rule in the propagation manner, the statuses of the nodes $v_6$, $v_5$, $v_3$, $v_2$, $v_1$, $\nu$ are changed to closed one after the other. By Theorem 3.2, $\Gamma$ is unsatisfiable. □

**Example 4.2.** Consider the following set $\Gamma$, which differs from the one in 4.1 in that $\leq 1000\, \sigma.(p \vee q)$ is replaced by $\leq 1001\, \sigma.(p \vee q)$:

$$\Gamma = \{\langle \sigma^* \rangle p,\ \neg p,\ [\sigma;\sigma;\sigma^*]\neg p,\ [\sigma](\neg p \vee \neg q),\ \geq 1000\, \sigma.q,\ \leq 1001\, \sigma.(p \vee q)\}.$$

A $\mathcal{C}_{\mathrm{GPDL}}$-tableau $G = (V, E, \nu)$ for $\Gamma$ is constructed analogously as in Example 4.1 up to the step 12, except that all the occurrences of "$\leq 1000$" are replaced by "$\leq 1001$". After that, the continuation is as follows. (In order to show that $\mathit{Status}(\nu)$ will never be changed to closed, we pay attention only to some paths of the tableau.)

13. Consider the moment when the statuses of the nodes $v_9$, $v_{11}$, $v_{12}$ and $v_{13}$ have been changed to closed. The set $\mathit{ILConstraints}(v_7)$ (which extends $\mathit{ILConstraints}_0(v_7)$ with $x_{v_i,e} = 0$ for $i \in \{9, 11, 12, 13\}$) is reduced to the following one w.r.t. feasibility:

$$\begin{array}{rcl} x_{v_8,\sigma} & \geq & 1 \\ x_{v_{10},\sigma} & \geq & 1000 \\ x_{v_8,\sigma} + x_{v_{10},\sigma} & \leq & 1001. \end{array}$$

14. Consider the node $v_8$.

- Applying $(\vee)$ to $v_8$ using the principal formula $p \vee q$, this node is connected to two new non-states $v_{27}$ and $v_{28}$ with

$$\begin{array}{rcl} \mathit{Label}(v_{27}) & = & \{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ \neg p \vee \neg q,\ p\} \\ \mathit{Label}(v_{28}) & = & \{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ \neg p \vee \neg q,\ q\}. \end{array}$$

- Applying $(\vee)$ to $v_{27}$ using the principal formula $\neg p \vee \neg q$, this node is connected to two new non-states $v_{29}$ and $v_{30}$ with

$$\begin{array}{rcl} \mathit{Label}(v_{29}) & = & \{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ \neg p,\ p\} \\ \mathit{Label}(v_{30}) & = & \{\langle A_1, 0\rangle p,\ [A_2, 1]\neg p,\ \neg q,\ p\}. \end{array}$$

- Applying ([A]) to $v_{30}$, this node is connected to a new non-state $v_{31}$ with $Label(v_{31}) = \{\langle A_1, 0\rangle p,\ [\sigma][A_2, 2]\neg p,\ \neg q,\ p\}$.
- Applying ($\langle \mathrm{A}\rangle_f$) to $v_{31}$, this node is connected to two new non-states $v_{32}$ and $v_{33}$ with

$$\begin{array}{rcl} Label(v_{32}) & = & \{[\sigma][A_2, 2]\neg p,\ \neg q,\ p\} \\ Label(v_{33}) & = & \{\langle\sigma\rangle\langle A_1, 0\rangle p,\ [\sigma][A_2, 2]\neg p,\ \neg q,\ p\}. \end{array}$$

- Applying ($form\text{-}state$) to $v_{32}$, this node is connected to a new state $v_{34}$ with $Label(v_{34}) = Label(v_{32})$. No tableau rule is applicable to $v_{34}$.

15. Consider the node $v_{10}$.

- Applying ($\vee$) to $v_{10}$ using the principal formula $p \vee q$, this node is connected to two new non-states $v_{35}$ and $v_{36}$ with

$$\begin{array}{rcl} Label(v_{35}) & = & \{q,\ [A_2, 1]\neg p,\ \neg p \vee \neg q,\ p\} \\ Label(v_{36}) & = & \{q,\ [A_2, 1]\neg p,\ \neg p \vee \neg q\}. \end{array}$$

- Applying ($\vee$) to $v_{36}$ using the principal formula $\neg p \vee \neg q$, this node is connected to two new non-states $v_{37}$ and $v_{38}$ with

$$\begin{array}{rcl} Label(v_{37}) & = & \{q,\ [A_2, 1]\neg p,\ \neg p\} \\ Label(v_{38}) & = & \{q,\ [A_2, 1]\neg p,\ \neg q\}. \end{array}$$

- Applying ([A]) to $v_{37}$, this node is connected to a new non-state $v_{39}$ with $Label(v_{39}) = \{q,\ [\sigma][A_2, 2]\neg p,\ \neg p\}$.
- Applying ($form\text{-}state$) to $v_{39}$, this node is connected to a new state $v_{40}$ with $Label(v_{40}) = Label(v_{39})$. No tableau rule is applicable to $v_{40}$.

16. Now, let's pay attention to the following paths in the constructed tableau, where only $v_7$ is a state:

- $\nu$, $v_1$, $v_2$, $v_3$, $v_5$, $v_6$, $v_7$,
- $v_7$, $v_8$, $v_{27}$, $v_{30}$, $v_{31}$, $v_{32}$, $v_{34}$,
- $v_7$, $v_{10}$, $v_{36}$, $v_{37}$, $v_{39}$, $v_{40}$.

Recall that no tableau rule is applicable to $v_{34}$ or $v_{40}$. Note that $ILConstraints(v_7) \cup \{x_{v_8,\sigma} \geq 1\}$ is the same as $ILConstraints(v_7)$, and hence $\langle\sigma\rangle\langle A_1, 0\rangle p$ is $\Diamond$-realizable at $v_7$. The status of any node in the above given paths cannot be changed to closed at any step. Therefore, at the end, we will have $Status(\nu) = \mathsf{expanded} \neq \mathsf{closed}$. By Theorem 3.2, $\Gamma$ is satisfiable. □

# 5. Conclusions

We have given the first direct tableau decision procedure for GPDL, which has EXPTIME (optimal) complexity when numbers are encoded in unary. It uses global caching and exploits our technique of integer linear feasibility checking [13, 16].

The condition 6 of Definition 3.1 of $\Diamond$-realizability is crucial for the combination of checking fulfillment of existential star modalities with integer linear feasibility checking. It is needed for the completeness of the tableau calculus. The proofs of soundness and completeness of the calculus required novel and advanced techniques, and really were a complicated task.

Our tableau calculus and decision procedure for GPDL have been designed to increase flexibility in choosing the control strategy and to simplify the presentation. For example, some essential conditions on applicability of the tableau rules are given, but a strict priority order of the rules is not. One can check inconsistencies caused by $\bot$ or a pair $\varphi$, $\overline{\varphi}$ at a node and propagate them on-the-fly, but check inconsistencies caused by infeasibility of a set of integer linear constraints or $\Diamond$-unrealizability only periodically. In general, our decision procedure is a framework, which can be implemented with various optimizations. The important points of the framework are the use of global caching and integer linear feasibility checking. Together, they guarantee the EXPTIME (optimal) complexity, and furthermore, the second one allows the use of highly optimized packages of integer linear programming (like IBM ILOG CPLEX Optimizer) and makes the task more scalable w.r.t. graded modalities.

As mentioned in the introduction, the method by De Giacomo and Lenzerini [10] for checking satisfiability in $\mathcal{CIQ}$ cannot give an efficient decision procedure for GPDL. Consider Example 4.1 with the occurrences of 1000 replaced by a constant natural number $n$. The method by De Giacomo and Lenzerini [10] uses a translation that, when applied to this example, results in a set of formulas with size $\Omega(n)$. In the case $n = 1000$, the problem resulting from the translation is already hard; in the case $n = 1000000$, it seems not practically solvable. When using our method for this example, as shown in Section 4, the sets $ILConstraints(v)$ that appear in the reasoning process are simple and cause no problems for highly optimized packages of integer linear programming like IBM ILOG CPLEX Optimizer.

We have implemented the reasoner TGC2 for reasoning in the description logics $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, $\mathcal{SHIO}$, the modal logic CPDL$_{reg}$ and some other modal/description logics.[5] Preliminary experiments with TGC2 showed that it deals with qualified number restrictions (graded modalities) in $\mathcal{SHIQ}$ much better than the well-known reasoners Racer, FaCT++, HermiT and Pellet. With respect to automated reasoning in GPDL, the current version 006.beta of TGC2 uses the decision procedure given in [12], which is incomplete. A correction for TGC2 in dealing with GPDL will be done soon (hopefully, before March 2016). We are not aware of any other reasoner for GPDL.

# References

[1] Fischer MJ, Ladner RE. Propositional Dynamic Logic of Regular Programs. J Comput Syst Sci. 1979;18(2):194–211.

[2] Harel D, Kozen D, Tiuryn J. Dynamic Logic. MIT Press; 2000.

[3] Dunin-Kęplicz B, Nguyen LA, Szałas A. Converse-PDL with regular inclusion axioms: a framework for MAS logics. Journal of Applied Non-Classical Logics. 2011;21(1):61–91.

[4] Giacomo GD, Lenzerini M. TBox and ABox Reasoning in Expressive Description Logics. In: Proceedings of KR'1996; 1996. p. 316–327.

[5] Pratt VR. A Near-Optimal Method for Reasoning about Action. J Comput Syst Sci. 1980;20(2):231–254.

[5] See `http://www.mimuw.edu.pl/~nguyen/TGC2`.

[6] Nguyen LA, Szałas A. Checking Consistency of an ABox w.r.t. Global Assumptions in PDL. Fundamenta Informaticae. 2010;102(1):97–113.

[7] Abate P, Goré R, Widmann F. An On-the-fly Tableau-based Decision Procedure for PDL-satisfiability. Electronic Notes in Theoretical Computer Science. 2009;231:191–209.

[8] Nguyen LA, Szałas A. An Optimal Tableau Decision Procedure for Converse-PDL. In: Nguyen NT, Bui TD, Szczerbicki E, Nguyen NB, editors. Proceedings of KSE'2009. IEEE Computer Society; 2009. p. 207–214.

[9] Goré R, Widmann F. Optimal and Cut-Free Tableaux for Propositional Dynamic Logic with Converse. In: Proceedings of IJCAR 2010. vol. 6173 of LNCS. Springer; 2010. p. 225–239.

[10] De Giacomo G, Lenzerini M. What's in an Aggregate: Foundations for Description Logics with Tuples and Sets. In: Proceedings of IJCAI 95. Morgan Kaufmann; 1995. p. 801–807.

[11] Tobies S. Complexity results and practical algorithms for logics in knowledge representation. RWTH-Aachen; 2001.

[12] Nguyen LA. ExpTime Tableaux with Global Caching for Graded Propositional Dynamic Logic. In: Proceedings of CS&P'2015. vol. 1492 of CEUR Workshop Proceedings. CEUR-WS.org; 2015. p. 44–56.

[13] Nguyen LA. ExpTime Tableaux for the Description Logic $\mathcal{SHIQ}$ Based on Global State Caching and Integer Linear Feasibility Checking; 2013. arXiv:1205.5838v4.

[14] De Giacomo G, Lenzerini M. Boosting the Correspondence between Description Logics and Propositional Dynamic Logics. In: Proceedings of AAAI 1994. AAAI Press / The MIT Press; 1994. p. 205–212.

[15] Nguyen LA. Designing a Tableau Reasoner for Description Logics. In: Proc. of ICCSAMA'2015. vol. 358 of Advances in Intelligent Systems and Computing. Springer; 2015. p. 321–333.

[16] Nguyen LA, Golińska-Pilarek J. An ExpTime Tableau Method for Dealing with Nominals and Qualified Number Restrictions in Deciding the Description Logic SHOQ. Fundam Inform. 2014;135(4):433–449.

# A. Complexity Analysis

Define the *length* of a formula (resp. program) to be the number bits used for the usual sequential representation of that formula (resp. program). Define the *size* of a set of formulas to be the sum of the lengths of its formulas. If $N$ is the size of $\Gamma$ and $\leq n\,\sigma.\varphi$ or $\geq n\,\sigma.\varphi$ occurs in $\Gamma$, then:

- when numbers are coded in unary we have $n < N$,
- when numbers are coded in binary we have $n < 2^N$.

**Definition A.1.** For a set $\Gamma$ of formulas in the base language, the set of *basic subformulas* of $\Gamma$, denoted by $bsf(\Gamma)$, consists of all formulas $\varphi$ and $\overline{\varphi}$ such that either $\varphi \in \Gamma$ or $\varphi$ is a subformula of a formula of $\Gamma$. The set $closure(\Gamma)$ is defined to be the smallest extension of $bsf(\Gamma)$ such that:

1. if $[\alpha]\varphi \in bsf(\Gamma)$, $q \in Q_{\mathbb{A}_\alpha}$, $\omega$ is of the form $\sigma$ or $\xi?$ and occurs in $\alpha$, then $[\mathbb{A}_\alpha, q]\varphi$ and $[\omega][\mathbb{A}_\alpha, q]\varphi$ belong to $closure(\Gamma)$;
2. if $\langle\alpha\rangle\varphi \in bsf(\Gamma)$, $q \in Q_{\mathbb{A}_\alpha}$, $\omega$ is of the form $\sigma$ or $\xi?$ and occurs in $\alpha$, then $\langle\mathbb{A}_\alpha, q\rangle\varphi$ and $\langle\omega\rangle\langle\mathbb{A}_\alpha, q\rangle\varphi$ belong to $closure(\Gamma)$;
3. if $(\geq n\,\sigma.\varphi) \in bsf(\Gamma)$, then $\langle\sigma\rangle\varphi \in closure(\Gamma)$;
4. if $\langle\sigma\rangle\varphi \in closure(\Gamma)$, then $(\geq 1\,\sigma.\varphi) \in closure(\Gamma)$. □

**Lemma A.2.** Formulas used for the construction of any $\mathcal{C}_{\mathrm{GPDL}}$-tableau for $\Gamma$ belong to $closure(\Gamma)$. The number of formulas of $closure(\Gamma)$ is of rank $O(N^3)$, where $N$ is the size of $\Gamma$.

**Proof:**
The first assertion is clear. Consider the second one. Observe that, if $\alpha$ is a program of length $n$, then we can construct $\mathbb{A}_\alpha$ with $O(n)$ states. Clearly, the size of $bsf(\Gamma)$ is of rank $O(N)$. Consider the construction of $closure(\Gamma)$ from $bsf(\Gamma)$. The first two rules of Definition A.1 add $O(N^3)$ formulas to $closure(\Gamma)$. The third rule of Definition A.1 add $O(N)$ formulas to $closure(\Gamma)$. The fourth rule of Definition A.1 add $O(N^3)$ formulas to $closure(\Gamma)$. Therefore, the number of formulas of $closure(\Gamma)$ is of rank $O(N^3)$. □

**Definition A.3.** An $\mathrm{IFDL}(l, m, n)$-problem (a problem of Linear Integer Feasibility for Description Logics with size specified by $l, m, n$) is specified as follows:

$$\sum_{j=1}^{m} a_{i,j} \cdot x_j \bowtie_i b_i, \quad \text{for } 1 \leq i \leq l;$$
$$x_j \geq 0, \quad \text{for } 1 \leq j \leq m;$$

where each $a_{i,j}$ is either 0 or 1, each $x_j$ is a variable standing for a natural number, each $\bowtie_i$ is either $\leq$ or $\geq$, each $b_i$ is a natural number encoded by using no more than $n$ bits (i.e., $b_i < 2^n$). The problem is *feasible* if it has a solution (i.e., values for the variables $x_j$, $1 \leq j \leq l$, that are natural numbers satisfying the constraints), and is *infeasible* otherwise. By solving an $\mathrm{IFDL}(l, m, n)$-problem we mean checking its feasibility. □

The following lemma was first given and proved in [13].

**Lemma A.4.** Every IFDL$(l, m, n)$-problem satisfying the following properties can be solved in (at most) exponential time in $n$:

- $l \leq n$, $m$ is (at most) exponential in $n$,
- and
  - either $b_i \leq n$ for all $1 \leq i \leq l$ such that $\bowtie_i$ is $\leq$
  - or $b_i \leq n$ for all $1 \leq i \leq l$ such that $\bowtie_i$ is $\geq$ □

**Lemma A.5.** Let $\Gamma$ be a finite set of formulas in NNF and $N$ the size of $\Gamma$. A $\mathcal{C}_{\mathrm{GPDL}}$-tableau for $\Gamma$ can be constructed in (at most) exponential time in $N$ in the following cases:

- numbers are coded in unary,
- numbers are coded in binary and, for each formula $\leq n\, \sigma.\varphi$ occurring in $\Gamma$, $n \leq N$,
- numbers are coded in binary and, for each formula $\geq n\, \sigma.\varphi$ occurring in $\Gamma$, $n \leq N$.

**Proof:**
Let's construct any $\mathcal{C}_{\mathrm{GPDL}}$-tableau $G = (V, E, \nu)$ for $\Gamma$. Recall that the number of formulas of $closure(\Gamma)$ is of rank $O(N^3)$. For each $v \in V$, both $Label(v)$ and $RFmls(v)$ are subsets of $closure(\Gamma)$. Since nodes of $G$ are globally cached (in the sense that, if $v_1$ and $v_2$ are different nodes, then $Type(v_1) \neq Type(v_2)$ or $Label(v_1) \neq Label(v_2)$) or $RFmls(v_1) \neq RFmls(v_2)$), it follows that $G$ has no more than $2^{N^6}$ nodes.

Checking feasibility of $ILConstraints(v)$ or $ILConstraints(v) \cup \{x_{w,\sigma} \geq 1\}$, where $v$ is a state, $w$ is a successor of $v$ and $\sigma \in ELabels(v, w)$, is an IFDL$(N, 2^{N^3} \times N, N)$-problem (since each successor created by the tableau rule $(trans)$ for a state is characterized by a pair $(\sigma, X)$ with $X \subseteq closure(\Gamma)$), which satisfies the assumptions of Lemma A.4 and can thus be solved in (at most) exponential time in $N$.

Observe that each of the following tasks can also be done in (at most) exponential time in $N$ (the multiplication of a polynomial of the number of nodes and the time for checking feasibility of a set of the form $ILConstraints(v)$ or $ILConstraints(v) \cup \{x_{w,\sigma} \geq 1\}$):

- a round of updating $\Diamond$-realizability and closedness for nodes,
- checking whether a rule is applicable to a fixed node and applying the rule to that node,
- choosing a rule applicable to some node and applying it to that node to make changes to the graph.

Since each node is expanded at most once and the status closed for nodes is permanent, we conclude that $\Gamma$ can be constructed in (at most) exponential time in $N$ for the cases mentioned in the lemma. □

# B. Proofs of Soundness and Completeness

In this section, let $\Gamma$ be a finite set of formulas in NNF and $G = (V, E, \nu)$ an arbitrary $\mathcal{C}_{\mathrm{GPDL}}$-tableau for $\Gamma$. We first define some auxiliary notions.

**Definition B.1.** Fix a sequential process of marking formulas of the form $\langle A, q\rangle\psi$ or $\langle\omega\rangle\langle A, q\rangle\psi$ in the labels of nodes of $G$ as $\Diamond$-realizable. For $v \in V$ and $\varphi \in \mathit{FullLabel}(v)$ of one of those forms, let $\mathit{TimeStampPR}(\varphi, v)$ denote the moment at which $\varphi$ is marked as $\Diamond$-realizable at $v$. □

**Definition B.2.** Let $v$ be a node of $G$ with $\mathit{Status}(v) \in \{\mathsf{unexpanded}, \mathsf{expanded}\}$ and let $\varphi \in \mathit{FullLabel}(v)$ be of the form $\langle A, q\rangle\psi$ or $\langle\omega\rangle\langle A, q\rangle\psi$. A *trace of $\Diamond$-realization* of $\varphi$ at $v$ is a sequence $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ such that:

- $k \geq 0$, $v_0 = v$ and $\varphi_0 = \varphi$,
- for each $0 \leq i \leq k$, $\varphi_i \in \mathit{FullLabel}(v_i)$, $\varphi_i$ is of the form $\langle A, q_i\rangle\psi$ or $\langle\omega_i\rangle\langle A, q_i\rangle\psi$ and is $\Diamond$-realizable at $v_i$,
- for each $0 \leq i < k$, $\mathit{TimeStampPR}(\varphi_i, v_i) > \mathit{TimeStampPR}(\varphi_{i+1}, v_{i+1})$ and $\varphi_i$ was marked (at the moment $\mathit{TimeStampPR}(\varphi_i, v_i)$) as $\Diamond$-realizable at $v_i$ due to the $\Diamond$-realizability of $\varphi_{i+1}$ at $v_{i+1}$ according to the rules specified by the conditions 2, 3, 4b–6 of Definition 3.1. □

**Definition B.3.** Let $v$ be a node of $G$ with $\mathit{Status}(v) \in \{\mathsf{unexpanded}, \mathsf{expanded}\}$ and let $\varphi \in \mathit{FullLabel}(v)$ be of the form $\langle A, q\rangle\psi$ or $\langle\omega\rangle\langle A, q\rangle\psi$. A *$\Diamond$-realization* of $\varphi$ at $v$ is a sequence $(v_0, \varphi_0), \ldots, (v_{k+1}, \varphi_{k+1})$ with $k \geq 0$ such that the sequence $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ is a trace of $\Diamond$-realization of $\varphi$ at $v$ and one of the following conditions holds:

- $\varphi_k$ is marked (at the moment $\mathit{TimeStampPR}(\varphi_k, v_k)$) as $\Diamond$-realizable at $v_k$ due to the rule specified by the condition 1 of Definition 3.1, i.e., $v_k$ is expanded by the tableau rule $(\langle \mathrm{A}\rangle_f)$ with $\varphi = \langle A, q_k\rangle\psi$ being the principal formula (having $q_k \in F_A$), and furthermore, $v_{k+1}$ is the successor of $v_k$ whose label is obtained from $\mathit{Label}(v_k)$ by replacing $\varphi_k$ by $\psi$, and $\mathit{Status}(v_{k+1}) \in \{\mathsf{unexpanded}, \mathsf{expanded}\}$;
- $\varphi_k$ is marked (at the moment $\mathit{TimeStampPR}(\varphi_k, v_k)$) as $\Diamond$-realizable at $v_k$ due to the rule specified by the condition 4a of Definition 3.1, i.e., $\varphi = \langle A, q_k\rangle\psi$, $q_k \in F_A$, $\psi \in \mathit{FullLabel}(v_k)$, and furthermore, $v_{k+1} = v_k$, $\varphi_{k+1} = \psi$ and $\mathit{Type}(v_{k+1}) = \mathsf{non\text{-}state}$. □

Note that, if $(v_0, \varphi_0), \ldots, (v_{k+1}, \varphi_{k+1})$ is a $\Diamond$-realization of $\varphi$ at $v$, then $\mathit{Type}(v_{k+1}) = \mathsf{non\text{-}state}$.

## B.1. Soundness

Recall that $\Gamma$ is a finite set of formulas in NNF and $G = (V, E, \nu)$ is an arbitrary $\mathcal{C}_{\mathrm{GPDL}}$-tableau for $\Gamma$.

**Definition B.4.** Let $\mathcal{M}$ be a Kripke model and $x \in \Gamma^{\mathcal{M}}$. A *marking* of $G$ w.r.t. $\mathcal{M}$ and $x$ is a structure $G' = (V', E', \nu', f, g, \mathit{ELabel})$, where $(V', E', \nu')$ is a (possibly infinite) tree with $V'$ being the set of nodes, $E'$ the set of edges, $\nu'$ the root, $f : V' \to V$, $g : V' \to \Delta^{\mathcal{M}}$, and $\mathit{ELabel}$ is a partial function from $E'$ to $\Sigma$, satisfying the following properties:

1. $f(\nu') = \nu$ and $g(\nu') = x$,
2. for every $v' \in V'$, $Label(f(v'))$ is satisfied at $g(v')$ in $\mathcal{M}$ (i.e., $g(v') \in (Label(f(v')))^{\mathcal{M}}$),
3. for every $(v', w') \in E'$, $(f(v'), f(w')) \in E$,
4. for every $(v', w') \in E'$ such that $Type(f(v')) = \mathsf{state}$, $ELabel(v', w') \in ELabels(f(v'), f(w'))$,
5. for every $v' \in V'$ such that $Type(f(v')) = \mathsf{non\text{-}state}$, there exists $w'$ such that $(v', w') \in E'$,
6. for every $v' \in V'$ such that $Type(f(v')) = \mathsf{state}$:
   - let $v = f(v')$ and $S$ be the assignment such that, for every $(v, w) \in E$ and every $\sigma \in ELabels(v, w)$, $S(x_{w,\sigma})$ is the number of $w'$ such that $(v', w') \in E'$, $f(w') = w$ and $ELabel(v', w') = \sigma$,
   - then $S$ is a solution of $ILConstraints_0(v)$,
7. for every $v' \in V'$ and every $\varphi \in FullLabel(f(v'))$ of the form $\langle A, q\rangle\psi$ of $\langle\omega\rangle\langle A, q\rangle\psi$, there exists a sequence $v'_0, \ldots, v'_k$ of nodes of $G'$ such that $v'_0 = v'$ and $\varphi$ has a $\Diamond$-realization of the form $(f(v'_0), \varphi_0), \ldots, (f(v'_k), \varphi_k)$ at $f(v')$ in $G$. □

**Lemma B.5.** Let $\mathcal{M}$ be a finite-branching Kripke model and $x \in \Gamma^{\mathcal{M}}$. If $G' = (V', E', \nu', f, g, ELabel)$ is a marking of $G$ w.r.t. $\mathcal{M}$ and $x$, then, for every $v' \in V'$, $Status(f(v')) \neq \mathsf{closed}$. □

This lemma holds because it can be proved in a straightforward way that, if $Status(f(v')) = \mathsf{closed}$, then $v' \notin V'$, by induction on the moment when $Status(f(v'))$ is changed to closed.

**Definition B.6.** Let $\mathcal{M}$ be a Kripke model of $\Gamma$, $\varphi = \langle A, q\rangle\psi$ and $x \in \varphi^{\mathcal{M}}$. We define the $\Diamond$-*rank* of $\varphi$ at $x$ to be the smallest natural number $k$ such that there exists a word $\omega_1 \ldots \omega_k$ accepted by $(A, q)$ with $(x, x_k) \in (\omega_1; \ldots; \omega_k)^M$ and $x_k \in \psi^M$. □

**Definition B.7.** Let $\mathcal{M}$ be a Kripke model of $\Gamma$, $\varphi = \langle\omega\rangle\langle A, q\rangle\psi$ and $x \in \varphi^{\mathcal{M}}$. We define the $\Diamond$-*rank* of $\varphi$ at $x$ to be $k + 1$, where $k$ the smallest natural number such that there exists $(x, y) \in \omega^{\mathcal{M}}$ such that $k$ is the $\Diamond$-rank of $\langle A, q\rangle\psi$ at $y$. □

**Lemma B.8.** Let $\mathcal{M}$ be a finitely-branching Kripke model, $x \in \Gamma^{\mathcal{M}}$ and $G' = (V', E', \nu', f, g, ELabel)$ the structure constructed by the function `CreateMarking`$(\Gamma, G, \mathcal{M}, x)$ (on page 21). Then $G'$ is a marking of $G$ w.r.t. $\mathcal{M}$ and $x$.

**Proof:**
The assertions 1–4 of Definition B.4 are easy to proved by induction on the moment when $v'$ is added to $V'$ or $(v', w')$ is added to $E'$. The assertion 5 of Definition B.4 holds due to the construction of $G'$. The assertion 7 of Definition B.4 holds due to the use of $\Diamond$-ranks in the construction of $G'$ and the fact that the tableau rule $(form\text{-}state)$ is applicable to a node $v$ only when no static tableau rule is applicable to $v$. Now consider the remaining assertion 6 of Definition B.4.

Consider a formula $(\geq n\, \sigma.\psi) \in Label(v)$ and the corresponding constraint $\sum\{x_{w,\sigma} \mid (v, w) \in E, \sigma \in ELabels(v, w), \psi \in Label(w)\} \geq n$ of $ILConstraints_0(v)$. Since $Label(v)$ is satisfied at $g(v')$ in

**Function** CreateMarking($\Gamma, G, \mathcal{M}, x$)

**Input**: a finite set $\Gamma$ of formulas in NNF, a $\mathcal{C}_{\mathrm{GPDL}}$-tableau $G = (V, E, \nu)$ for $\Gamma$, a finitely-branching Kripke model $\mathcal{M}$ and $x \in \Gamma^{\mathcal{M}}$.

**Output**: a marking $G' = (V', E', \nu', f, g, ELabel)$ of $G$ w.r.t. $\mathcal{M}$ and $x$.

1 $V' := \{\nu'\}$, $E' := \emptyset$, set $ELabel$ to the empty partial mapping;
2 initialize $f$ and $g$ by $f(\nu') := \nu$ and $g(\nu') := x$;
3 set $unsolved$ to the queue containing only $\nu'$;
4 **while** *$unsolved$ is not empty* **do**
5  extract $v'$ from $unsolved$ and set $v := f(v')$;
6  **if** *$v$ was expanded by the tableau rule $(form\text{-}state)$ or a static tableau rule such that the principal formula is not of the form $\langle A, q\rangle\psi$* **then**
7  let $w$ be a successor of $v$ in $G$ such that $g(v') \in (Label(w))^{\mathcal{M}}$;
8  add a new $w'$ to $V'$ and $unsolved$, add $(v', w')$ to $E'$, set $f(w') := w$ and $g(w') := g(v')$;
9  **if** *$v$ was expanded by a static tableau rule with $\varphi = \langle A, q\rangle\psi$ being the principal formula* **then**
10  let $w$ be a successor of $v$ in $G$ such that, if $\varphi'$ is the formula in $Label(w)$ that is obtained from $\varphi$, then $g(v') \in (\varphi')^{\mathcal{M}}$ and either $\varphi' = \psi$ or the $\Diamond$-rank of $\varphi'$ at $g(v')$ is less than the $\Diamond$-rank of $\varphi$ at $g(v')$;
11  add a new $w'$ to $V'$ and $unsolved$, add $(v', w')$ to $E'$, set $f(w') := w$ and $g(w') := g(v')$;
12  **if** *$v$ was expanded by the tableau rule $(trans)$* **then**
13  **foreach** *$y \in \Delta^{\mathcal{I}}$ and each $\sigma \in \Sigma$ such that $(g(v'), y) \in \sigma^{\mathcal{M}}$* **do**
14  **if** *there exist successors $w$ of $v$ in $G$ such that $y \in (Label(w))^{\mathcal{M}}$ and $\sigma \in ELabels(v, w)$* **then**
15  among those successors choose $w$ such that $Label(w)$ is maximal (w.r.t. $\subseteq$);
16  add a new $w'$ to $V'$ and $unsolved$, set $f(w') := w$ and $g(w') := y$;
17  add $(v', w')$ to $E'$ and set $ELabel(v', w') := \sigma$;

$\mathcal{M}$, there are $n$ pairwise different $y_1, \ldots, y_n$ such that $(g(v'), y_i) \in \sigma^{\mathcal{M}}$ and $y_i \in \psi^{\mathcal{M}}$ for all $1 \leq i \leq n$. For each $y \in \{y_1, \ldots, y_n\}$, there exists a successor $w$ of $v$ in $G$ specified at the step 15 of the procedure `CreateMarking` with the property that $\sigma \in ELabels(v, w)$ and $y \in Label(w))^{\mathcal{M}}$, and a new $w'$ is created with $(v', w') \in E'$, $f(w') = w$ and $ELabel(v', w') = \sigma$. Hence, the mentioned constraint is satisfied by $S$.

Consider a formula $(\leq n\, \sigma.\psi) \in Label(v)$ and the corresponding constraint $\sum\{x_{w,\sigma} \mid (v, w) \in E, \sigma \in ELabels(v, w), \psi \in Label(w)\} \leq n$ of $ILConstraints_0(v)$. Observe that, if $(v, w) \in E$, $\sigma \in ELabels(v, w)$, $\psi \in Label(w)$ and $x_{w,\sigma} > 0$, then, by the definition of $S$, there exists $w'$ such that $(v', w') \in E'$, $f(w') = w$ and $ELabel(v', w') = \sigma$, and due to the creation of $w'$, there must exist $y$ such that $(g(v'), y) \in \sigma^{\mathcal{M}}$ and $y \in (Label(w))^{\mathcal{M}}$, which implies $y \in \psi^{\mathcal{M}}$. Let $y_1, \ldots, y_m$ be all pairwise different elements such that $(g(v'), y_i) \in \sigma^{\mathcal{M}}$ and $y_i \in \psi^{\mathcal{M}}$ for $1 \leq i \leq m$. Since $(\leq n\, \sigma.\psi) \in Label(v)$ and $Label(v)$ is satisfied at $g(v')$ in $\mathcal{M}$, we have $m \leq n$. For each $y \in \{y_1, \ldots, y_m\}$, there exists a unique successor $w$ of $v$ in $G$ specified at the step 15 of the procedure `CreateMarking` with the property that $\sigma \in ELabels(v, w)$ and $y \in (Label(w))^{\mathcal{M}}$, and exactly one $w'$ is created with $(v', w') \in E'$, $f(w') = w$

and $ELabel(v', w') = \sigma$. Hence, $\sum\{S(x_{w,\sigma}) \mid (v, w) \in E, \sigma \in ELabels(v, w), \psi \in Label(w)\} = m$, and consequently, the mentioned constraint is satisfied by $S$.

We have shown that $S$ is a solution of $ILConstraints_0(v)$, which completes the proof. □

**Corollary B.9. (Soundness)**
Let $\Gamma$ be a finite set of formulas in NNF and $G = (V, E, \nu)$ an arbitrary $\mathcal{C}_{\mathrm{GPDL}}$-tableau for $\Gamma$. If $\Gamma$ is satisfiable, then $Status(\nu) \neq \mathsf{closed}$.

**Proof:**
Assume that $\Gamma$ is satisfiable. It is well known that, if a set of formula is satisfiable, then it is has a finitely-branching Kripke model. Let $\mathcal{M}$ be a finitely-branching Kripke model of $\Gamma$ and let $x \in \Gamma^{\mathcal{M}}$. Let $G' = (V', E', \nu', f, g, ELabel)$ be the structure constructed by the function `CreateMarking`$(\Gamma, G, \mathcal{M}, x)$. By Lemma B.8, $G'$ is a marking of $G$ w.r.t. $\mathcal{M}$ and $x$, and by Lemma B.5, $Status(f(\nu')) \neq \mathsf{closed}$, which means $Status(\nu) \neq \mathsf{closed}$. □

## B.2. Completeness

In this subsection, assume that $Status(\nu) \neq \mathsf{closed}$ (where $\nu$ is the root of $G$). We prove that $\Gamma$ is satisfiable by constructing a *selected unfolding* $G'$ of $G$, the *model tree* $G''$ of $G'$, and the Kripke model $\mathcal{M}$ that corresponds to $G''$.

**Definition B.10.** Given a non-state $v \in V$ with $Status(v) \neq \mathsf{closed}$, a *saturation path* of $v$ is a sequence $v_0, v_1, \ldots, v_k$ of nodes of $G$, with $k \geq 1$, such that:

- $v_0 = v$,
- for every $0 \leq i < k$, $Type(v_i) = \mathsf{non\text{-}state}$, $Status(v_i) \neq \mathsf{closed}$ and $(v_i, v_{i+1}) \in E$,
- $Type(v_k) = \mathsf{state}$ and $Status(v_k) \neq \mathsf{closed}$. □

Observe that each saturation path of $v$ is finite.[6] Furthermore, if $v_i$ is a non-state with $Status(v_i) \neq \mathsf{closed}$ then $v_i$ has a successor $v_{i+1}$ with $Status(v_{i+1}) \neq \mathsf{closed}$. Therefore, we have the following lemma.

**Lemma B.11.** If $Type(v) = \mathsf{non\text{-}state}$ and $Status(v) \neq \mathsf{closed}$, then $v$ has at least one saturation path.

**Definition B.12.** A *selective unfolding* of $G$ is a structure $G' = (V', E', \nu', f)$, where $(V', E', \nu')$ is a (possibly infinite) tree with $V'$ being the set of nodes, $E'$ the set of edges, $\nu'$ the root, and $f : V' \to V$, with the following properties:

1. $f(\nu') = \nu$,
2. if $v' \in V'$, then $Status(f(v')) \neq \mathsf{closed}$,
3. if $(v', w') \in E'$, then $(f(v'), f(w')) \in E$,

[6] If a non-state $v_{i+1}$ is a successor of a non-state $v_i$ then $RFmls(v_{i+1}) \supset RFmls(v_i)$. Also recall that $RFmls(v_{i+1})$ is a subset of the finite set $closure(\Gamma)$.

4. if $v' \in V'$ and $\mathit{Type}(f(v')) = \mathsf{non\text{-}state}$, then there exists a unique $w' \in V'$ such that $(v', w') \in E'$, and furthermore, $(f(v'), f(w')) \in E$,

5. if $v' \in V'$, $v = f(v')$ and $\mathit{Type}(v) = \mathsf{state}$, then there exists a solution $S$ of $\mathit{ILConstraints}(v)$ such that:[7]

   (a) suppose $w_1, \ldots, w_n$ ($n \geq 0$) are all the (pairwise different) successors of $v$ in $G$ such that, for $1 \leq i \leq n$, there exists $\sigma_i \in \mathit{ELabels}(v, w_i)$ such that $S(x_{w_i,\sigma_i}) \geq 1$,

   (b) then $v'$ has exactly $n$ (pairwise different) successors $w'_1, \ldots, w'_n$ in $G'$ (with $(v', w'_i) \in E'$ for $1 \leq i \leq n$), which satisfy $f(w'_i) = w_i$ for $1 \leq i \leq n$. □

We want to construct a selective unfolding $G' = (V', E', \nu', f)$ of $G$ with the following property:

for every $v' \in V'$ such that $\mathit{Type}(v) = \mathsf{state}$, where $v = f(v')$, and for every $\varphi = \langle\sigma\rangle\langle A, q\rangle\psi \in \mathit{Label}(v)$, there exist a sequence $v'_0, \ldots, v'_k$ of nodes of $G'$ and a $\Diamond$-realization $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ of $\varphi$ at $v$ in $G$ such that $v_i = f(v'_i)$ for $0 \leq i \leq k$, and in the case $i \geq 1$, if $v_i = v_{i-1}$, then $v'_i = v'_{i-1}$, else $(v'_{i-1}, v'_i) \in E'$. (2)

The construction is specified by the following definition.

**Definition B.13.** A *selected unfolding* of $G$ is a selective unfolding $G' = (V', E', \nu', f)$ of $G$ constructed as follows:

1. $V' := \{\nu'\}$, $E' := \emptyset$, $\mathit{unrealized} := \emptyset$, $\mathit{unexpanded} := \emptyset$,

2. initialize $f$ by setting $f(\nu') := \nu$,

3. `Expand` $(\nu')$ (defined on page 24),

4. while $\mathit{unrealized} \neq \emptyset$ or $\mathit{unexpanded} \neq \emptyset$ do:

   (a) choose $v' \in V'$ with a minimal depth in the tree $G'$ (as in breadth-first search) such that either $v' \in \mathit{unexpanded}$ or $(v', \varphi) \in \mathit{unrealized}$ for some $\varphi$ (and choose such a $\varphi$),

   (b) if $v' \in \mathit{unexpanded}$ then `Expand` $(v')$,

   (c) else `Realize` $(v', \varphi)$ (defined on page 24). □

**Lemma B.14.** The variables $\mathit{unrealized}$ and $\mathit{unexpanded}$ used in the construction specified in Definition B.13 satisfy the following properties:

- $\mathit{unexpanded} \subseteq V'$ and, if $v' \in \mathit{unexpanded}$ and $f(v') = v$, then $\mathit{Type}(v) = \mathsf{state}$;

- $\mathit{unrealized}$ contains pairs $(v', \varphi)$ such that $v' \in V'$, $\mathit{Type}(f(v')) = \mathsf{state}$, $\varphi$ belongs to $\mathit{Label}(f(v'))$ and is of the form $\langle\sigma\rangle\langle A, q\rangle\psi$.

Furthermore, as an invariant of the main loop of that construction, $G'$ is a selective unfolding of $G$. □

[7] $S$ is a value assignment for the variables occurring in $\mathit{ILConstraints}(v)$.

**Procedure** Expand($v'$)

**Global data**: a finite selective unfolding $G' = (V', E', \nu', f)$ of $G$ and the sets $unrealized$, $unexpanded$.

**Input**: a node $v' \in V'$ such that, if $Type(f(v')) = \mathsf{state}$, then $v' \in unexpanded$.

let $v = f(v')$;
**if** $Type(v) = \mathsf{non\text{-}state}$ **then**
  let $v'_0 = v'$ and let $v_0, \ldots, v_k$ be a saturation path of $v$;
  **foreach** $1 \leq i \leq k$ **do**
    add a new node $v'_i$ to $V'$ and the edge $(v'_{i-1}, v'_i)$ to $E'$;
    $f(v'_i) := v_i$;
  add $v'_k$ to $unexpanded$;
  **foreach** $\varphi \in Label(v_k)$ *such that* $\varphi$ *is of the form* $\langle\sigma\rangle\langle A, q\rangle\psi$ **do**
    add the pair $(v'_k, \varphi)$ to $unrealized$;
**else if** *there exists* $\varphi$ *such that* $(v', \varphi) \in unrealized$ **then**
  `ExpandAndRealize` $(v', \varphi)$; `// defined on page 25`
**else**
  let $S$ be a solution of $ILConstraints(v)$ and let $w_1, \ldots, w_k$ be all the successors of $v$ such that, for $1 \leq i \leq k$, there exists $\sigma_i \in ELabels(v, w_i)$ such that $S(x_{w_i,\sigma_i}) \geq 1$;
  **foreach** $1 \leq i \leq k$ **do**
    add a new node $w'_i$ to $V'$ and the edge $(v', w'_i)$ to $E'$;
    $f(w'_i) := w_i$ and `Expand` $(w'_i)$; `// we have` $Type(w_i) = \mathsf{non\text{-}state}$
  delete $v'$ from $unexpanded$;

**Procedure** Realize($v', \varphi$)

**Global data**: a finite selective unfolding $G' = (V', E', \nu', f)$ of $G$ and the sets $unrealized$, $unexpanded$.

**Input**: a pair $(v', \varphi) \in unrealized$ with $v' \notin unexpanded$.

let $v = f(v')$ and let $v'_0, \ldots, v'_k$ be a sequence of nodes of $G'$ such that:

- $v'_0 = v'$ and, for $0 \leq i < k$, either $(v'_i, v'_{i+1}) \in E'$ or $v'_i = v'_{i+1}$; and
- either there exists a $\Diamond$-realization $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ of $\varphi$ at $v$ in $G$ such that $v_i = f(v'_i)$ for $0 \leq i \leq k$,
- or $v'_k$ is a leaf of $G'$ (i.e., without any $v'_{k+1}$ such that $(v'_k, v'_{k+1}) \in E'$) and there exists a trace $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ of $\Diamond$-realization of $\varphi$ at $v$ in $G$ such that $v_i = f(v'_i)$ for $0 \leq i \leq k$;

**if** $v'_k$ *is a leaf of* $G'$ **then** `ExpandAndRealize` $(v'_k, \varphi_k)$; `// defined on page 25`

The proof of this lemma is straightforward.

The construction specified in Definition B.13 uses the procedures `Expand`$(v')$, `Realize`$(v', \varphi)$ and `ExpandAndRealize`$(v', \varphi)$. Clearly, at any time during the construction, $G'$ is a finite structure. The resulting $G'$, however, may be infinite. To show that the mentioned procedures are well defined, we

**Procedure** ExpandAndRealize($v', \varphi$)

**Global data**: a finite selective unfolding $G' = (V', E', \nu', f)$ of $G$ and the sets $unrealized$, $unexpanded$.

**Input**: a pair $(v', \varphi) \in unrealized$ with $v' \in unexpanded \subseteq V'$.

1 let $v = f(v')$ and let $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ be a $\Diamond$-realization of $\varphi$ at $v$;
2 $v'_0 := v'$;
3 **foreach** $0 \leq i < k$ **do**
4  **if** $Type(v_i) =$ non-state **then**
5  **if** $v_{i+1} = v_i$ **then** $v'_{i+1} := v'_i$
6  **else**
7  add a new node $v'_{i+1}$ to $V'$ and the edge $(v'_i, v'_{i+1})$ to $E'$;
8  $f(v'_{i+1}) := v_{i+1}$;
9  **else**
10  **if** $i > 0$ **then**
11  **foreach** $\varphi' \in Label(v_i)$ *such that* $\varphi'$ *is of the form* $\langle\sigma'\rangle\langle A', q'\rangle\psi'$ **do**
12  add the pair $(v'_i, \varphi')$ to $unrealized$;
13  let $S$ be a solution of $ILConstraints(v_i) \cup \{x_{v_{i+1},\sigma_i} \geq 1\}$, where $\varphi_i = \langle\sigma_i\rangle\langle A, q_i\rangle\psi$, and let $v_{i+1,1}, \ldots, v_{i+1,j_i}$ be all the successors of $v_i$ in $G$ such that, for $1 \leq j \leq j_i$, there exists $\sigma_{i,j} \in ELabels(v_i, v_{i+1,j})$ such that $S(x_{v_{i+1,j},\sigma_{i,j}}) \geq 1$;
14  let $l_i$ be the index among $1, \ldots, j_i$ such that $v_{i+1} = v_{i+1,l_i}$;
15  **foreach** $1 \leq j \leq j_i$ **do**
16  add a new node $v'_{i+1,j}$ to $V'$ and the edge $(v'_i, v'_{i+1,j})$ to $E'$;
17  $f(v'_{i+1,j}) := v_{i+1,j}$; // we have $Type(v_{i+1,j}) =$ non-state
18  **if** $j \neq l_i$ **then** Expand $(v'_{i+1,j})$ **else** $v'_{i+1} := v'_{i+1,j}$;
19 Expand $(v'_k)$ and delete $v'$ from $unexpanded$;

justify the "let" statements occurring in them:

- The statement "let $v_0, \ldots, v_k$ be a saturation path of $v$" at step 3 of the procedure Expand is well specified due to Lemma B.11.
- The statement "let $S$ be a solution of $ILConstraints(v)$" at step 13 of the procedure Expand is well specified because $Type(v) =$ state, $Status(v) \neq$ closed and due to the condition 3 of the tableau rule $(close)$.
- Consider the statement "let $v'_0, \ldots, v'_k$ be a sequence of nodes of $G'$ such that [...]" at step 1 of the procedure Realize($v', \varphi$). The sequence can be computed as follows:
  - $h := 0$, $v'_0 := v'$, $v_0 := f(v'_0)$, $\varphi_0 := \varphi$,
  - while $v'_h$ is not a leaf of $G'$ do:
    - (invariant: $(v_0, \varphi_0), \ldots, (v_h, \varphi_h)$ is a prefix of a $\Diamond$-realization of $\varphi$ at $v$ in $G$)

- if $Type(v_h) = \mathsf{non\text{-}state}$ then:
  - set $w'$ to the unique successor of $v'_h$ in $G'$ and set $w := f(w')$ (we have that $w$ is a successor of $v_h$ in $G$),
  - if there exists a $\Diamond$-trace $(v_h, \varphi_h), \ldots, (v_{h+l}, \varphi_{h+l})$ of $\varphi_h$ at $v_h$ in $G$ such that $v_h = \ldots = v_{h+l}$, then set $v'_{h+1}, \ldots, v'_{h+l}$ to $v'_h$, $k := h + l$ and break the loop,
  - else there must exist a formula in $FullLabel(w)$ that is obtained from $\varphi_h$ via a trace $(v_h, \varphi_h), \ldots, (v_{h+l}, \varphi_{h+l})$ of $\Diamond$-realization of $\varphi_h$ at $v_h$ in $G$, with $v_h = \ldots = v_{h+l-1}$ and $v_{h+l} = w$, and we thus set $v'_{h+1}, \ldots, v'_{h+l-1}$ to $v'_h$, $v'_{h+l} := w'$ and $h := h + l$,
- else, due to the invariant, $\varphi_h$ must be of the form $\langle \sigma_h \rangle \langle A, q_h \rangle \psi$, hence, by the tableau rule $(\Diamond_{\geq})$, $(\geq 1\, \sigma_h.\langle A, q_h \rangle \psi) \in Label(v_h)$, and consequently, there exist a successor $v_{h+1}$ of $v_h$ in $G$ and a successor $v'_{h+1}$ of $v'_h$ in $G'$ such that $\sigma_h \in ELabels(v_h, v_{h+1})$, $\langle A, q_h \rangle \psi \in Label(v_{h+1})$ and $f(v'_{h+1}) = v_{h+1}$, and we thus set $\varphi_{h+1} := \langle A, q_h \rangle \psi$ and $h := h + 1$.

- The statement "let $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ be a $\Diamond$-realization of $\varphi$ at $v$" at the step 1 of the procedure `ExpandAndRealize`$(v', \varphi)$ is well specified due to Lemma B.14.

- The statement "let $S$ be a solution of $ILConstraints(v_i) \cup \{x_{v_{i+1},\sigma_i} \geq 1\}$, where $\varphi_i = \langle \sigma_i \rangle \langle A, q_i \rangle \psi$" at the step 13 of the procedure `ExpandAndRealize`$(v', \varphi)$ is well specified due to the condition 6 of Definition 3.1.

**Lemma B.15.** The notion of selected unfolding is well defined and any selected unfolding $G'$ of $G$ satisfies the expected property (2). □

The first assertion of this lemma follows from the above given observations and Lemma B.14. The second assertion follows from the construction of $G'$.

**Definition B.16.** Let $G' = (V', E', \nu', f)$ be a selected unfolding of $G$. The *model tree* of $G'$ is a structure $G'' = (V'', E'', \nu'', g, Label, ELabel)$, where $(V'', E'', \nu'')$ is a (possibly infinite) tree with $V''$ being the set of nodes, $E''$ the set of edges, $\nu''$ the root, $g : V'' \to V'$, $ELabel : E'' \to \Sigma$, and $Label$ is a mapping that associates each $v'' \in V''$ with a set of formulas, constructed as follows:

- $V'' := \{\nu''\}$, $E'' := \emptyset$ and set $g$, $Label$, $ELabel$ to empty mappings.

- Let $v'_0, \ldots, v'_h$ be the branch in $G'$ starting from $v'_0 = \nu'$ such that $Type(f(v'_0)) = \ldots = Type(f(v'_{h-1})) = \mathsf{non\text{-}state}$ and $Type(f(v'_h)) = \mathsf{state}$. Set $g(\nu'') := v'_h$ and $Label(\nu'') := FullLabel(f(v'_{h-1}))$.

- For each $v'' \in V''$ do:

  - Let $v' = g(v'')$ and $w'_1, \ldots, w'_k$ be all the (pairwise different) successors of $v'$ in $G'$. (As an invariant, we have $Type(f(v')) = \mathsf{state}$.)
  - Let $v = f(v')$ and $w_i = f(w'_i)$ for $1 \leq i \leq k$. Let $S_{v'}$ be the solution of $ILConstraints(v)$ that was used for creating $w'_1, \ldots, w'_k$ in the construction of $G'$.[8]

[8] as in the step 13 of the procedure `Expand` or in the step 13 of the procedure `ExpandAndRealize`

– For each $1 \leq i \leq k$, each $\sigma \in ELabels(v, w_i)$ such that $S_{v'}(x_{w_i,\sigma}) \geq 1$, and each $1 \leq j \leq S_{v'}(x_{w_i,\sigma})$, do:
  * Let $u'_0, \ldots, u'_m$ be the branch in $G'$ starting from $u'_0 = w'_i$ such that $Type(f(u'_0)) = \ldots = Type(f(u'_{m-1})) = \mathsf{non\text{-}state}$ and $Type(f(u'_m)) = \mathsf{state}$. (The branch does not depend on $\sigma$ and $j$.)
  * Add a new node $w''$ to $V''$ and the edge $(v'', w'')$ to $E''$.
  * Set $g(w'') := u'_m$, $ELabel(v'', w'') := \sigma$, $Label(w'') := FullLabel(f(u'_{m-1}))$. □

Roughly speaking, the model tree of $G'$ is obtained from $G'$ by:

- cloning successors of each node $v'$ such that $Type(f(v')) = \mathsf{state}$ appropriately, according to the corresponding solution of $ILConstraints(f(v'))$,
- merging nodes on each branch fragment in $G'$ that corresponds to a saturation path in $G$.

As shown by the lemma given below, the model tree of $G'$ is similar to a Hintikka structure.

**Lemma B.17.** Let $G' = (V', E', \nu', f)$ be a selected unfolding of $G$ and $G'' = (V'', E'', \nu'', g, Label, ELabel)$ the model tree of $G'$. Then, for every $v'' \in V''$ and every $\varphi \in Label(v'')$:

1. $\bot \notin Label(v'')$ and $\overline{\varphi} \notin Label(v'')$,
2. if $\varphi = \psi \wedge \xi$, then $\{\psi, \xi\} \subset Label(v'')$,
3. if $\varphi = \psi \vee \xi$, then $\psi \in Label(v'')$ or $\xi \in Label(v'')$,
4. if $\varphi = [\alpha]\psi$, $\alpha \notin \Sigma$, $\alpha$ is not a test and $I_{\mathbf{A}_\alpha} = \{q_1, \ldots, q_k\}$, then $\{[\mathbf{A}_\sigma, q_1]\psi, \ldots, [\mathbf{A}_\sigma, q_k]\psi\} \subset Label(v'')$,
5. if $\varphi = [A, q]\psi$ and $\delta_A(q) = \{(\omega_1, q_1), \ldots, (\omega_k, q_k)\}$, then $\{[\omega_1][A, q_1]\psi, \ldots, [\omega_k][A, q_k]\psi\} \subset Label(v'')$,
6. if $\varphi = [A, q]\psi$ and $q \in F_A$, then $\psi \in Label(v'')$,
7. if $\varphi = [\xi?]\psi$, then $\overline{\xi} \in Label(v'')$ or $\psi \in Label(v'')$,
8. if $\varphi = [\sigma]\psi$, $(v'', w'') \in E''$ and $ELabel(v'', w'') = \sigma$, then $\psi \in Label(w'')$,
9. if $\varphi = \langle\alpha\rangle\psi$, $\alpha \notin \Sigma$, $\alpha$ is not a test and $I_{\mathbb{A}_\alpha} = \{q_1, \ldots, q_k\}$, then some of the formulas $\langle\mathbb{A}_\alpha, q_1\rangle\psi$, $\ldots$, $\langle\mathbb{A}_\alpha, q_k\rangle\psi$ belong to $Label(v'')$,
10. if $\varphi = \langle\xi?\rangle\psi$, then $\{\xi, \psi\} \subseteq Label(v'')$,
11. if $\varphi = \langle\sigma\rangle\psi$, then there exists $w''$ such that $(v'', w'') \in E''$, $ELabel(v'', w'') = \sigma$ and $\psi \in Label(w'')$,
12. if $\varphi = \langle A, q\rangle\psi$, then there exist a 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 $w''_0, \ldots, w''_k$ of nodes of $G''$ such that $w''_0 = v''$, $\psi \in Label(w''_k)$ and, for each $1 \leq i \leq k$, if $\omega_i = (\xi_i?)$, then $w''_i = w''_{i-1}$ and $\{\xi_i, \langle A, q_i\rangle\psi\} \subseteq Label(w''_i)$, else $(w''_{i-1}, w''_i) \in E''$, $ELabel(w''_{i-1}, w''_i) = \omega_i$ and $\langle A, q_i\rangle\psi \in Label(w''_i)$,

13. if $\varphi = (\geq n\, \sigma.\psi)$, then there are at least $n$ successors $w''_1, \ldots, w''_n$ of $v''$ in $G''$ such that $ELabel(v'', w''_i) = \sigma$ and $\psi \in Label(w''_i)$ for all $1 \leq i \leq n$,

14. if $\varphi = (\leq n\, \sigma.\psi)$, then there are at most $n$ successors $w''_1, \ldots, w''_n$ of $v''$ in $G''$ such that $ELabel(v'', w''_i) = \sigma$ and $\psi \in Label(w''_i)$ for all $1 \leq i \leq n$,

15. if $\varphi = (\leq n\, \sigma.\psi)$, $(v'', w'') \in E''$ and $ELabel(v'', w'') = \sigma$, then either $\psi \in Label(w'')$ or $\overline{\psi} \in Label(w'')$.

**Proof:**
This lemma follows from the construction and properties of $G$, $G'$ and $G''$. More concrete indications are as follows. The first assertion holds due to the fact that, for every $v' \in V'$, $Status(f(v')) \neq \mathsf{closed}$. The assertions 2–7, 9, 10 hold due to the static tableau rules. The assertion 8 holds due the tableau rule $(trans)$. The assertion 11 holds due the tableau rules $(\Diamond_{\geq})$ and $(trans)$. The assertion 12 holds due Lemma B.15 and the fact that, for every $v' \in V'$, $Status(f(v')) \neq \mathsf{closed}$. The assertions 13-15 hold due the tableau rule $(trans)$. □

**Definition B.18.** Let $G' = (V', E', \nu', f)$ be a selected unfolding of $G$ and $G'' = (V'', E'', \nu'', g, Label, ELabel)$ the model tree of $G'$. The *Kripke model corresponding to* $G''$ is $\mathcal{M} = (\Delta^{\mathcal{M}}, \cdot^{\mathcal{M}})$ defined as follows: $\Delta^{\mathcal{M}} = V''$, $p^{\mathcal{M}} = \{v'' \in V'' \mid p \in Label(v'')\}$ for $p \in \mathcal{PROP}$, and $\sigma^{\mathcal{M}} = \{(v'', w'') \in E'' \mid ELabel(v'', w'') = \sigma\}$ for $\sigma \in \Sigma$. □

**Lemma B.19.** Let $G' = (V', E', \nu', f)$ be a selected unfolding of $G$, $G'' = (V'', E'', \nu'', g, Label, ELabel)$ the model tree of $G'$, and $\mathcal{M}$ the Kripke model corresponding to $G''$. Then, for every $v'' \in \Delta^{\mathcal{M}}$ and every $\varphi \in Label(v'')$, we have $v'' \in \varphi^{\mathcal{M}}$. □

This lemma can be proved by induction on the structure of $\varphi$ in a straightforward way, using Lemma B.17.

**Corollary B.20. (Completeness)**
Let $\Gamma$ be a finite set of formulas in NNF and $G = (V, E, \nu)$ an arbitrary $\mathcal{C}_{\mathrm{GPDL}}$-tableau for $\Gamma$. If $Status(\nu) \neq \mathsf{closed}$, then $\Gamma$ is satisfiable. □

This corollary follows from Lemma B.19 and the fact that $\Gamma \subseteq Label(\nu'')$, where $\nu''$ is the root of $G''$ mentioned in Lemma B.19.