Checking Consistency of an ABox w.r.t. Global Assumptions in PDL*

Linh Anh Nguyen and Andrzej Szałas

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

Abstract. We reformulate Pratt's tableau decision procedure of checking satisfiability of a set of formulas in PDL. Our formulation is simpler and more direct for implementation. Extending the method we give the first ExpTime (optimal) tableau decision procedure not based on transformation for checking consistency of an ABox w.r.t. a TBox in PDL (here, PDL is treated as a description logic). We also prove the new result that the data complexity of the instance checking problem in PDL is coNP-complete.

1 Introduction

Propositional dynamic logic (PDL) is a multi-modal logic introduced by Fischer and Ladner [8] for reasoning about programs. It is useful not only for program verification but also for other fields of computer science like knowledge representation and artificial intelligence (e.g., [14, 15, 4, 6]). For example, the description logic $\mathcal{ALC}_{reg}$, a notational variant of PDL, can be used for reasoning about structured knowledge.

The problem of checking satisfiability of a set of formulas in PDL is ExpTime-complete. This result was established by Fischer and Ladner [8], but their decision procedure for PDL is via filtration and canonical model and therefore is not really practical. The first practical and optimal (ExpTime) decision procedure for PDL was given by Pratt [24]. The essence of his procedure is based on constructing an "and-or" graph for the considered set of formulas by using tableau rules and global caching, and then checking whether a model for the set can be extracted from the graph. However, the formulation of his procedure is a bit too indirect: it goes via a labeled tableau calculus, tree-like labeled tableaux, tree-like traditional ("lean") tableaux, and "and-or" graphs.

De Giacomo and Massacci [5] gave a NExpTime algorithm for checking satisfiability in CPDL (i.e., PDL with converse) and informally described how to transform the algorithm to an ExpTime version. Abate et al. [1] gave a "single-pass" tableau decision procedure for checking satisfiability in PDL. Their algorithm does not exploit global caching [24, 12] and has complexity 2ExpTime in the worst cases. Recently, independently with this paper[1], Goré and Widmann improved the algorithm given in [1] using global caching to obtain an ExpTime tableau decision procedure for the satisfiability problem of PDL [13]. However, proofs of correctness of their algorithm are not available to us at the moment. There are a few prototype implementations for checking satisfiability in PDL [27, 17, 1, 13].

There is a tight relationship between multi-modal logics and description logics which will often be exploited in this paper. Two basic components of description logic theories are ABoxes and TBoxes. An ABox (*assertion box*) consists of facts and a TBox (*terminological box*) consists of formulas expressing relationships between concepts. Two basic reasoning problems considered in description logics, amongst others, are:

1. the problem of checking consistency of an ABox w.r.t. a TBox,
2. the instance checking problem.

The first tableau-based procedure for $\mathcal{ALC}_{reg}$ (PDL) in the description logic context was proposed by Baader [2] (the correspondence between $\mathcal{ALC}_{reg}$ and PDL had not yet been known). His procedure, however, has non-optimal complexity 2ExpTime. The correspondence between description logics like $\mathcal{ALC}_{reg}$ and PDL was first described in Schild's paper [26]. In [9], encoding the ABox

* Supported by the MNiSW grant N N206 399334.

[1] The first version [21] of this paper has been available since April 2009.

by "nominals" and "internalizing" the TBox, De Giacomo showed that the complexity of checking consistency of an ABox w.r.t. a TBox in CPDL is ExpTime-complete. In [10], using a transformation that encodes the ABox by a concept assertion plus terminology axioms, De Giacomo and Lenzerini showed that the mentioned problem is also ExpTime-complete for the description logic $\mathcal{CIQ}$ (an extension of CPDL).

In this paper, we reformulate Pratt's algorithm of checking satisfiability of a set of formulas in PDL. Our formulation is directly based on building an "and-or" graph by using traditional (unlabeled) tableau rules and global caching and is therefore simpler and more direct for implementation. Extending the method we give the *first* ExpTime (optimal) tableau decision procedure not based on transformation (encoding) for checking consistency of an ABox w.r.t. a TBox in PDL.

Despite that the upper-bound ExpTime is known for the complexity of the mentioned satisfiability problem in CPDL, implemented tableau provers for description logics usually have non-optimal complexity 2ExpTime. In the overview [3], Baader and Sattler wrote: *"The point in designing these [non-optimal] algorithms was not to prove worst-case complexity results, but ... to obtain 'practical' algorithms ... that are easy to implement and optimise, and which behave well on realistic knowledge bases. Nevertheless, the fact that 'natural' tableau algorithms for such* ExpTime*-complete logics are usually* NExpTime*-algorithms is an unpleasant phenomenon. ... Attempts to design* ExpTime*-tableaux for such logics (De Giacomo et al., 1996; De Giacomo and Massacci, 1996; Donini and Massacci, 1999) usually lead to rather complicated (and thus not easy to implement) algorithms, which (to the best of our knowledge) have not been implemented yet."* [3, page 26].

Our formulation of tableau calculi and decision procedures for PDL is short and clear, which makes the procedures natural and easy to implement. The first author has implemented a tableau prover called TGC for the basic description logic $\mathcal{ALC}$, which is also based on "and-or" graphs with global caching. The test results of TGC on the sets T98-sat and T98-kb of DL'98 Systems Comparison are comparable with the test results of the best systems DLP-98 and FaCT-98 that took part in that comparison (see [18]). One can say that the mentioned test sets are not representative for practical applications, but the comparison at least shows that optimization techniques can be applied (not only for $\mathcal{ALC}$ but also PDL) to obtain decision procedures that are both efficient in practice and optimal w.r.t. complexity.

We also study the data complexity of the instance checking problem in PDL. For the well-known description logic $\mathcal{SHIQ}$, Hustadt et al. [16] proved that the data complexity of that problem is coNP-complete. The lower bound for the data complexity of that problem in PDL ($\mathcal{ALC}_{reg}$) is known to be coNP-hard (shown for $\mathcal{ALC}$ by Schaerf in [25]). In this paper, by establishing the upper bound, we prove the new result that the data complexity of the instance checking problem in PDL is coNP-complete.

The rest of this paper is structured as follows. In Section 2, we define syntax and semantics of PDL. In Section 3 we formulate the problems we deal with. In Section 4, we present a tableau calculus for checking satisfiability of a set of formulas w.r.t. a set of global assumptions in PDL. In Section 5, we extend that calculus for checking consistency of an ABox w.r.t. a set of global assumptions (i.e., a TBox) in PDL. In Section 6, we give decision procedures based on our tableau calculi for the mentioned problems and derive the data complexity result. Conclusions are given in Section 7. Due to the lack of space, proofs of soundness and completeness of the calculi are contained in the full version [21] of this paper.

2 Propositional Dynamic Logic

We use Π_0 to denote the set of *atomic programs*, and Φ_0 to denote the set of *propositions* (i.e., atomic formulas). We denote elements of Π_0 by letters like σ, and elements of Φ_0 by letters like p and q.

A *Kripke model* is a pair $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$, where $\Delta^{\mathcal{M}}$ is a set of *states*, and $\cdot^{\mathcal{M}}$ is an interpretation function that maps each proposition p to a subset $p^{\mathcal{M}}$ of $\Delta^{\mathcal{M}}$, and each atomic program σ to a binary relation $\sigma^{\mathcal{M}}$ on $\Delta^{\mathcal{M}}$. Intuitively, for $p \in \Phi_0$, $p^{\mathcal{M}}$ is the set of states, where p is true, and for $\sigma \in \Pi_0$, $\sigma^{\mathcal{M}}$ is interpreted as a binary relation consisting of pairs (input_state, output_state).

Formulas and *programs* of PDL are defined respectively by the following BNF grammar rules:

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

We use letters like α, β to denote programs, and φ, ψ, ξ to denote formulas.

The intended meaning of program operators is the following:

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

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

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

$$\top^{\mathcal{M}} = \Delta^{\mathcal{M}} \qquad \bot^{\mathcal{M}} = \emptyset$$
$$(\neg\varphi)^{\mathcal{M}} = \Delta^{\mathcal{M}} \setminus \varphi^{\mathcal{M}} \qquad (\varphi \rightarrow \psi)^{\mathcal{M}} = (\neg\varphi \vee \psi)^{\mathcal{M}}$$
$$(\varphi \wedge \psi)^{\mathcal{M}} = \varphi^{\mathcal{M}} \cap \psi^{\mathcal{M}} \qquad (\varphi \vee \psi)^{\mathcal{M}} = \varphi^{\mathcal{M}} \cup \psi^{\mathcal{M}}$$

$$(\langle\alpha\rangle\varphi)^{\mathcal{M}} = \{x \in \Delta^{\mathcal{M}} \mid \exists y(\alpha^{\mathcal{M}}(x,y) \wedge \varphi^{\mathcal{M}}(y))\}$$
$$([\alpha]\varphi)^{\mathcal{M}} = \{x \in \Delta^{\mathcal{M}} \mid \forall y(\alpha^{\mathcal{M}}(x,y) \rightarrow \varphi^{\mathcal{M}}(y))\}$$

$$(\alpha;\beta)^{\mathcal{M}} = \alpha^{\mathcal{M}} \circ \beta^{\mathcal{M}} \qquad (\alpha \cup \beta)^{\mathcal{M}} = \alpha^{\mathcal{M}} \cup \beta^{\mathcal{M}}$$
$$(\alpha^*)^{\mathcal{M}} = (\alpha^{\mathcal{M}})^* \qquad (\varphi?)^{\mathcal{M}} = \{(x,x) \mid \varphi^{\mathcal{M}}(x)\}$$

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

3 The Problems We Address

When interpreting PDL as a description logic, states in a Kripke model, formulas, and programs are regarded respectively as "objects", "concepts", and "roles". A finite set Γ of global assumptions is treated as a "TBox". As for description logics, we introduce ABoxes and consider the problem of checking whether a given ABox is consistent with a given TBox, which is related to the instance checking problem.

We prefer to use the terminology of PDL instead of that of $\mathcal{ALC}_{reg}$ because this work is related to Pratt's work on PDL. We use the term *state variable* as an equivalent for the term "individual" used in description logic, and use letters like a, b, c to denote state variables. We extend the notion of Kripke model so that the interpretation function $\cdot^{\mathcal{M}}$ of a Kripke model $\mathcal{M}$ maps each state variable a to a state $a^{\mathcal{M}}$ of $\mathcal{M}$.

An *ABox* is a finite set of *assertions* of the form $a:\varphi$ or $\sigma(a,b)$, where φ is a formula in negation normal form (NNF) and a, b are state variables.[2] The meaning of $a:\varphi$ is that formula φ is satisfied in state a. An ABox is *extensionally reduced* if it contains only assertions of the form $a:p$ or $\sigma(a,b)$. We will refer to ABox assertions also as formulas. When necessary, we refer to formulas that are not ABox assertions as *traditional formulas*.

A *TBox* is a finite set of traditional formulas in NNF.

A Kripke model $\mathcal{M}$ *satisfies* an ABox $\mathcal{A}$ if $a^{\mathcal{M}} \in \varphi^{\mathcal{M}}$ for all $(a:\varphi) \in \mathcal{A}$ and $(a^{\mathcal{M}}, b^{\mathcal{M}}) \in \sigma^{\mathcal{M}}$ for all $\sigma(a,b) \in \mathcal{A}$. An ABox $\mathcal{A}$ is *satisfiable w.r.t.* (or *consistent with*) a TBox Γ iff there exists a Kripke model $\mathcal{M}$ that satisfies $\mathcal{A}$ and validates Γ.

[2] In NNF, the connective $\rightarrow$ does not occur and $\neg$ occurs only immediately before propositions. Every formula can be transformed to an equivalent formula in NNF.

The first problem we address is the problem of checking satisfiability of an ABox w.r.t. a TBox

Consider the use of PDL as a description logic. A pair $(\mathcal{A}, \Gamma)$ of an ABox $\mathcal{A}$ and a TBox Γ is treated as a knowledge base. A Kripke model that satisfies $\mathcal{A}$ and validates Γ is called a model of $(\mathcal{A}, \Gamma)$. Given a (traditional) formula φ (treated as a "concept") and a state variable a (treated as an "individual"), the problem of checking whether $a^{\mathcal{M}} \in \varphi^{\mathcal{M}}$ in every model $\mathcal{M}$ of $(\mathcal{A}, \Gamma)$ is called the instance checking problem (in PDL).

The second problem considered in this paper is the instance checking problem. The condition to check is denoted in such cases by $(\mathcal{A}, \Gamma) \models \varphi(a)$.

4 A Tableau Calculus for PDL

In this section, we do not consider ABoxes yet, and by a "formula" we mean a "traditional formula". Let X and Γ be finite sets of formulas. Consider the problem of checking whether X is satisfiable in PDL w.r.t. the set Γ of global assumptions. We assume that formulas are represented in NNF. We write $\overline{\varphi}$ to denote the NNF of $\neg\varphi$.

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

Our calculus $\mathcal{C}_{\mathrm{PDL}}$ will be specified as a finite set of tableau rules, which are used to expand nodes of "and-or" graphs. A *tableau rule* is specified with the following informations:

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

Usually, a tableau rule is written downwards, with a set of formulas above the line as the *premise*, which represents the label of the node to which the rule is applied, and a number of sets of formulas below the line as the *(possible) conclusions*, which represent the labels of the successor nodes resulting from the application of the rule. Possible conclusions of an "or"-rule are separated by |, while conclusions of an "and"-rule are separated/specified using &. If a rule is a unary rule (i.e. a rule with only one possible conclusion) or an "and"-rule then its conclusions are "firm" and we ignore the word "possible". An "or"-rule has the meaning that, if the premise is satisfiable w.r.t. Γ then some of the possible conclusions is also satisfiable w.r.t. Γ. On the other hand, an "and"-rule has the meaning that, if the premise is satisfiable w.r.t. Γ then all of the conclusions are also satisfiable w.r.t. Γ (possibly in different states of the model under construction). Note that, apart from the labels, there are also sets $rfs(\_)$ to be specified for the successor nodes.

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

Define tableau calculus $\mathcal{C}_{\mathrm{PDL}}$ w.r.t. a set Γ of global assumptions to be the set of the tableau rules given in Table 1. The rule $(trans)$ is the only "and"-rule and the only *transitional rule*. Instantiating this rule, for example, to $Y = \{\langle\sigma\rangle p, \langle\sigma\rangle q, [\sigma]r\}$ and $\Gamma = \{s\}$ we get two conclusions: $\{p, r, s\}$ and $\{q, r, s\}$. The other rules of $\mathcal{C}_{\mathrm{PDL}}$ are "or"-rules, which are also called *static rules*. The intuition of the sorting of static/transitional is that the static rules keep us in the same state of the model under construction, while each conclusion of the transitional rule takes us to a new state. For any rule of $\mathcal{C}_{\mathrm{PDL}}$ except $(trans)$, the distinguished formulas of the premise are called the *principal formulas* of the rule. The principal formulas of the rule $(trans)$ are the formulas of the form $\langle\sigma\rangle\varphi$ of the premise. We assume that any of the rules $(\wedge)$, $(\vee)$, $(\Box_;)$, $(\Box_\cup)$, $(\Box_?)$, $(\Box_*)$ is applicable to a node v only when the principal formula does not belong to $rfs(v)$. Applying a static rule different from $(\bot_0)$ and $(\bot)$ to a node v, for any successor node w of v, let $rfs(w)$ be the set that extends $rfs(v)$ with the principal formula of the applied rule. Applying any other rule to a node v, for any successor node w of v, let $rfs(w) = \emptyset$. Recall that $rfs(w)$ stands for "the set of formulas that have been reduced by a static rule after the last application of the transitional rule".

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

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

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

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

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

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

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

Table 1. Rules of the tableau calculus $\mathcal{C}_{\mathrm{PDL}}$

Observe that, by using $rfs(\_)$ and the restriction on applicability of the rules $(\wedge)$, $(\vee)$, $(\Box_;)$, $(\Box_\cup)$, $(\Box_?)$, and $(\Box_*)$, in any sequence of applications of static rules a formula of the form $\varphi \wedge \psi$, $\varphi \vee \psi$, $[\alpha;\beta]\varphi$, $[\alpha \cup \beta]\varphi$, $[\psi?]\varphi$, or $[\alpha^*]\varphi$ is reduced (as a principal formula) at most once. We do not adopt such a restriction for the rules $(\Diamond_;)$, $(\Diamond_\cup)$, $(\Diamond_?)$, and $(\Diamond_*)$ because we will require formulas of the form $\langle\alpha\rangle\varphi$ to be "realized" (in a finite number of steps).

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

An *"and-or" graph for* (X, Γ), also called a *tableau for* (X, Γ), is an "and-or" graph defined as follows. The initial node ν of the graph, called the *root* of the graph, is specified by $\mathcal{L}(\nu) = X \cup \Gamma$ and $rfs(\nu) = \emptyset$. For every node v of the graph, if a tableau rule of $\mathcal{C}_{\mathrm{PDL}}$ is applicable to the label of v in the sense that an instance of the rule has $\mathcal{L}(v)$ as the premise and $Z_1, \ldots, Z_k$ as the possible conclusions, then choose such a rule accordingly to the preference[3] and apply it to v to create k successors $w_1, \ldots, w_k$ of v with $\mathcal{L}(w_i) = Z_i$ for $1 \leq i \leq k$. If the graph already contains a node w_i' with the same contents as w_i then instead of creating a new node w_i as a successor of v we just connect v to w_i' and assume $w_i = w_i'$. If the applied rule is $(trans)$ then we *label* the edge (v, w_i) by the principal formula corresponding to the successor w_i. If the rule expanding v is an "or"-rule then v is an *"or"-node*, else v is an *"and"-node*. The information about which rule is applied to v is recorded for later uses. If no rule is applicable to v then v is an *end node*. Note that each node is "expanded" only once (using one rule). Also note that the graph is constructed using *global caching* [24, 11, 12] and the contents of its nodes are unique.

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

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

Let G be an "and-or" graph for (X, Γ), G' a marking of G, v a node of G', and $\langle\alpha\rangle\varphi$ a formula of the label of v. A *trace* of $\langle\alpha\rangle\varphi$ in G' starting from v is a sequence $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ such that:[4]

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

[4] This definition of trace is inspired by [23].

– $v_0 = v$ and $\varphi_0 = \langle\alpha\rangle\varphi$;
– for every $1 \leq i \leq k$, (v_{i-1}, v_i) is an edge of G';
– for every $1 \leq i \leq k$, φ_i is a formula of the label of v_i such that: if φ_{i-1} is not a principal formula of the tableau rule expanding v_{i-1}, then the rule must be a static rule and $\varphi_i = \varphi_{i-1}$, else
  - if the rule is $(\Diamond_;)$, $(\Diamond_\cup)$ or $(\Diamond_*)$ then φ_i is the formula obtained from φ_{i-1},
  - if the rule is $(\Diamond_?)$ and $\varphi_{i-1} = \langle\psi?\rangle\xi$ then $\varphi_i = \xi$,
  - else the rule is $(trans)$, φ_{i-1} is of the form $\langle\sigma\rangle\xi$ and is the label of the edge (v_{i-1}, v_i), and $\varphi_i = \xi$.

A trace $(v_0, \varphi_0), \ldots, (v_k, \varphi_k)$ of $\langle\alpha\rangle\varphi$ in G' is called a $\Diamond$-*realization* in G' for $\langle\alpha\rangle\varphi$ at v_0 if $\varphi_k = \varphi$.
A marking G' of an "and-or" graph G for (X, Γ) is *consistent* if:

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

Theorem 4.1 (Soundness and Completeness of $\mathcal{C}_{\mathrm{PDL}}$). *Let X and Γ be finite sets of formulas in NNF, and G be an "and-or" graph for (X, Γ). Then X is satisfiable w.r.t. the set Γ of global assumptions iff G has a consistent marking.* ◁

See the full version [21] for the proof of this theorem and for an example of "and-or" graph.

5 A Tableau Calculus for Dealing with ABoxes

Define tableau calculus $\mathcal{C}_{\mathrm{PDL+ABox}}$ w.r.t. a TBox Γ to be the extension of $\mathcal{C}_{\mathrm{PDL}}$ with the following additional rules:

– a rule (ρ') obtained from each rule $(\rho) \in \{(\wedge), (\vee), (\Box_;), (\Diamond_;), (\Box_\cup), (\Diamond_\cup), (\Box_?), (\Diamond_?), (\Box_*), (\Diamond_*)\}$ by labeling the principal formula and the formulas obtained from it by prefix "$a:$ " and adding the modified principal formula to each of the possible conclusions; for example:

$$(\vee') \ \frac{Y,\ a : \varphi \vee \psi}{Y,\ a : \varphi \vee \psi,\ a : \varphi \mid Y,\ a : \varphi \vee \psi,\ a : \psi}$$

– and

$$(\bot'_0) \ \frac{Y,\ a : \bot}{\bot} \qquad (\bot') \ \frac{Y,\ a : p,\ a : \neg p}{\bot}$$

$$(\Box') \ \frac{Y,\ a : [\sigma]\varphi,\ \sigma(a, b)}{Y,\ a : [\sigma]\varphi,\ \sigma(a, b),\ b : \varphi}$$

$$(trans') \ \frac{Y}{\&\{\ (\{\varphi\} \cup \{\psi \text{ s.t. } (a : [\sigma]\psi) \in Y\} \cup \Gamma) \text{ s.t. } (a : \langle\sigma\rangle\varphi) \in Y\ \}}$$

The additional rules of $\mathcal{C}_{\mathrm{PDL+ABox}}$ work on sets of ABox assertions, except that the conclusions of $(trans')$ are sets of traditional formulas. That is, in those rules, Y denotes a set of ABox assertions. The rule $(trans')$ is an "and"-rule and a transitional rule. The other additional rules of $\mathcal{C}_{\mathrm{PDL+ABox}}$ are "or"-rules and static rules.

Note that, for any additional static rule of $\mathcal{C}_{\mathrm{PDL+ABox}}$ except $(\bot'_0)$ and $(\bot')$, the premise is a subset of each of the possible conclusions. Such rules are said to be *monotonic*.

We assume that any of the rules $(\wedge')$, $(\vee')$, $(\Box'_;)$, $(\Diamond'_;)$, $(\Box'_\cup)$, $(\Diamond'_\cup)$, $(\Box'_?)$, $(\Diamond'_?)$, $(\Box'_*)$, $(\Diamond'_*)$ is applicable to a node v only when the principal formula does not belong to $rfs(v)$. Applying any of these rules to a node v, for any successor node w of v, let $rfs(w)$ be the set that extends $rfs(v)$ with the principal formula of the applied rule. We assume that the rule $(\Box')$ is applicable only when its conclusion is a proper superset of its premise. Applying this rule to a node v, let $rfs(w) = rfs(v)$ for the successor w of v. Applying $(\bot'_0)$, $(\bot')$, or $(trans')$ a node v, for any successor node w of v, let $rfs(w) = \emptyset$.

Similarly as for $\mathcal{C}_{\text{PDL}}$, we assume the following preference for the rules of $\mathcal{C}_{\text{PDL+ABox}}$: the rules $(\bot_0)$, $(\bot)$, $(\bot'_0)$, $(\bot')$ have the highest priority; unary static rules have a higher priority that non-unary static rules; all the static rules have a higher priority than the transitional rules.

Consider the problem of checking whether a given ABox $\mathcal{A}$ is satisfiable w.r.t. a given TBox Γ. We construct an *"and-or" graph for* $(\mathcal{A}, \Gamma)$ as follows. The graph will contain nodes of two kinds: *complex nodes* and *simple nodes*. The sets $\mathcal{L}(v)$ and $rfs(v)$ of a complex node v consist of ABox assertions, while such sets of a simple node v consist of traditional formulas. The graph will never contain edges from a simple node to a complex node. The root of the graph is a complex node ν with $\mathcal{L}(\nu) = \mathcal{A} \cup \{(a:\varphi) \mid \varphi \in \Gamma \text{ and } a \text{ is a state variable occurring in } \mathcal{A}\}$ and $rfs(\nu) = \emptyset$. Complex nodes are expanded using the additional rules of $\mathcal{C}_{\text{PDL+ABox}}$ (i.e., the "prime" rules), while simple nodes are expanded using the rules of $\mathcal{C}_{\text{PDL}}$. The "and-or" graph is expanded in the same way as described in the previous section for checking consistency of a set X of traditional formulas w.r.t. Γ.

The notion of *marking* remains unchanged.

Let G be an "and-or" graph for $(\mathcal{A}, \Gamma)$ and let G' be a marking of G. If v is a simple node of G' and $\langle\alpha\rangle\varphi$ is a formula of the label of v then a *trace* of $\langle\alpha\rangle\varphi$ in G' starting from v is defined as before. Consider the case when v is a complex "and"-node and suppose that $a:\langle\alpha\rangle\varphi \in \mathcal{L}(v)$. A *static trace* of $a:\langle\alpha\rangle\varphi$ at v is a sequence $\varphi_0, \ldots, \varphi_k$ such that:

- $\varphi_0 = \langle\alpha\rangle\varphi$;
- for every $1 \leq i \leq k$, $(a:\varphi_i) \in \mathcal{L}(v)$;
- for every $1 \leq i \leq k$,
  - if $\varphi_{i-1} = \langle\beta;\gamma\rangle\psi$ then $\varphi_i = \langle\beta\rangle\langle\gamma\rangle\psi$,
  - if $\varphi_{i-1} = \langle\beta \cup \gamma\rangle\psi$ then φ_i is either $\langle\beta\rangle\psi$ or $\langle\gamma\rangle\psi$,
  - if $\varphi_{i-1} = \langle\psi?\rangle\xi$ then $\varphi_i = \xi$,
  - if $\varphi_{i-1} = \langle\beta^*\rangle\psi$ then φ_i is either ψ or $\langle\beta\rangle\langle\beta^*\rangle\psi$.

A static trace $\varphi_0, \ldots, \varphi_k$ of $a:\langle\alpha\rangle\varphi$ at v is called a *static realization* for $a:\langle\alpha\rangle\varphi$ at v if either $\varphi_k = \varphi$ or φ_k is of the form $\langle\sigma\rangle\varphi'_k$ for some $\sigma \in \Pi_0$.

A marking G' of an "and-or" graph G for $(\mathcal{A}, \Gamma)$ is *consistent* if:

local consistency: G' does not contain any node with label $\{\bot\}$;
global consistency:

- for every complex "and"-node v of G', every formula of the form $a:\langle\alpha\rangle\varphi$ of the label of v has a static realization (at v),
- for every simple node v of G', every formula of the form $\langle\alpha\rangle\varphi$ of the label of v has a $\Diamond$-realization (starting at v) in G'.

Theorem 5.1 (Soundness and Completeness of $\mathcal{C}_{\text{PDL+ABox}}$). *Let $\mathcal{A}$ be an ABox, Γ a TBox, and G an "and-or" graph for $(\mathcal{A}, \Gamma)$. Then $\mathcal{A}$ is satisfiable w.r.t. Γ iff G has a consistent marking.*◁

See the full version [21] for the proof of this theorem and for an example of "and-or" graph involving ABoxes.

6 Decision Procedures for PDL

In this section, we present simple algorithms for checking satisfiability of a given set X of traditional formulas w.r.t. a given set Γ of global assumptions and for checking satisfiability of given ABox $\mathcal{A}$ w.r.t. a given TBox Γ. (Optimizations for the algorithms are discussed in [21].) We also prove the mentioned data complexity result for PDL.

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

6.1 Checking Satisfiability of X w.r.t. Γ

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

- Nodes of G_t are pairs (v, φ), where v is a node of G' and φ is a formula of the label of v.

– A pair $((v,\varphi),(w,\psi))$ is an edge of G_t if v is a node of G', φ is of the form $\langle\alpha\rangle\xi$, and the sequence (v,φ), (w,ψ) is a trace of φ in G'.

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

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

Theorem 6.1. *Let X and Γ denote finite sets of traditional formulas in NNF. Algorithm 1 is an* ExpTime *decision procedure for checking whether X is satisfiable w.r.t. the set Γ of global assumptions.*

Proof. It is easy to show that the algorithm has the invariant that a consistent marking of G cannot contain any node of $UnsatNodes$. The algorithm returns *false* only when the root v_0 belongs to $UnsatNodes$, that is, only when G does not have any consistent marking. At Step 7b, G' is a marking of G that satisfies the local consistency property. If at that step $V = \emptyset$ then it satisfies also the global consistency property and is thus a consistent marking of G. That is, the algorithm returns *true* only when G has a consistent marking. Therefore, by Theorem 4.1, Algorithm 1 is a decision procedure for the considered problem. See the full version [21] for an analysis of the complexity of the algorithm. ◁

6.2 Checking Satisfiability of an ABox w.r.t. a TBox

Let $\mathcal{A}$ be an ABox, Γ be a TBox, G be an "and-or" graph for $(\mathcal{A},\Gamma)$, and G' be a marking of G. The *graph G_t of traces of G' in G* is the largest graph such that:

– A node of G_t is
  - either a pair (v,φ), where v is a simple node of G' and $\varphi \in \mathcal{L}(v)$,
  - or a pair $(v, a{:}\varphi)$, where v is a complex "and"-node of G' and $(a{:}\varphi) \in \mathcal{L}(v)$.
– An edge of G_t is
  - either a pair $((v,\varphi),(w,\psi))$ such that v is a simple node of G', φ is of the form $\langle\alpha\rangle\xi$, and the sequence (v,φ), (w,ψ) is a trace of φ in G',
  - or a pair $((v, a:\varphi),(v, a:\psi))$ such that v is a complex "and"-node of G', φ is of the form $\langle\alpha\rangle\xi$, and the sequence φ, ψ is a static trace of $a{:}\varphi$ at v,
  - or a pair $((v, a:\langle\sigma\rangle\varphi),(w,\varphi))$ such that v is a complex "and"-node of G' and (v,w) is an edge of G' with $a{:}\langle\sigma\rangle\varphi$ as the label.

A node of G_t is an *end node* if it is of the form (v,φ) or $(v, a{:}\varphi)$ such that φ is not of the form $\langle\alpha\rangle\xi$. A node of G_t is *productive* if there is a path connecting it to an end node.

By *Algorithm 1′* we refer to the algorithm obtained from Algorithm 1 by changing X to $\mathcal{A}$ and modifying Step 7a to "let V be the set of all nodes v of G' such that G_t contains a non-productive node of the form $(v,\langle\alpha\rangle\varphi)$ or $(v, a{:}\langle\alpha\rangle\varphi)$". Algorithm 1′ receives an ABox $\mathcal{A}$ and a TBox Γ as input and checks whether $\mathcal{A}$ is satisfiable w.r.t. Γ.

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

1. construct an "and-or" graph G with root v_0 for (X, Γ);
2. $UnsatNodes := \emptyset$;
3. if G contains a node v with label $\{\bot\}$ then
   $updateUnsatNodes(G, UnsatNodes, \{v\})$;
4. if $v_0 \in UnsatNodes$ then return *false*;
5. let G' be the maximal subgraph of G without nodes from $UnsatNodes$;
   (we have that G' is a marking of G)
6. construct the graph G_t of traces of G' in G;
7. while $v_0 \notin UnsatNodes$ do:
   (a) let V be the set of all nodes v of G' such that G_t contains a non-productive node of the form $(v, \langle\alpha\rangle\varphi)$;
   (b) if $V = \emptyset$ then return *true*;
   (c) $updateUnsatNodes(G, UnsatNodes, V)$;
   (d) if $v_0 \in UnsatNodes$ then return *false*;
   (e) let G' be the maximal subgraph of G without nodes from $UnsatNodes$;
   (we have that G' is a marking of G)
   (f) update G_t to the graph of traces of G' in G;

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

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

Fig. 1. Algorithm for checking satisfiability of X w.r.t. Γ.

Lemma 6.2. *Let $\mathcal{A}$ be an ABox, Γ be a TBox, G be an "and-or" graph for $(\mathcal{A}, \Gamma)$, and n be the size of $\mathcal{A} \cup \Gamma$. Then G has $2^{O(n^2)}$ nodes. If v is a simple node of G then $\mathcal{L}(v)$ and $rfs(v)$ contain at most $O(n)$ formulas and are of size $O(n^2)$. If v is a complex node of G then $\mathcal{L}(v)$ and $rfs(v)$ contain at most $O(n^2)$ formulas and are of size $O(n^3)$.*

See [21] for the proof of this lemma. Using this lemma and the proof of Theorem 6.1 with appropriate changes (see also [21, Lemma 6.2]) we obtain the following theorem.

Theorem 6.3. *Algorithm 1′ is an* ExpTime *decision procedure for checking whether a given ABox $\mathcal{A}$ is satisfiable w.r.t. a given TBox Γ.* ◁

Algorithm 1′ uses global caching for both complex nodes and simple nodes. What happens if we use global caching only for simple nodes and backtracking on branchings at complex "or"-nodes? Is the complexity still ExpTime? The rest of this subsection deals with these questions.

Lemma 6.4. *Let $\mathcal{A}$ be an ABox, Γ be a TBox, and G be an "and-or" graph for $(\mathcal{A}, \Gamma)$. Then G has a consistent marking iff there exists a complex "and"-node v of G such that the subgraph generated by v of G (which uses v as the root) has a consistent marking.*

Proof. Just notice that the root of G is a complex node and every father node of a complex node must be a complex "or"-node. ◁

By *Algorithm 1″* we refer to the algorithm that checks whether a given ABox $\mathcal{A}$ is satisfiable w.r.t. a given TBox Γ as follows. The algorithm "simulates" the tasks of constructing an "and-or" graph for $(\mathcal{A}, \Gamma)$ and checking whether the graph has a consistent marking but does it as follows:

1. nondeterministically expand a path from the root until reaching a complex "and"-node v;
2. construct the full subgraph rooted at v;
3. check whether the subgraph has a consistent marking (as done in the steps 2–7 of Algorithm 1), and return *true* if it does;
4. if none of the possible executions returns *true* then return *false*.

In practice, the first step of the above algorithm is executed by backtracking on the branchings of the applications of "or"-rules. The algorithm does not keep all complex nodes but only the ones on the current path of complex nodes. On the other hand, simple nodes can be globally cached. That is, simple nodes can be left through backtracking for use in the next possible executions.

Theorem 6.5. *Using backtracking to deal with nondeterminism, Algorithm 1″ is an* ExpTime *decision procedure for checking whether a given ABox $\mathcal{A}$ is satisfiable w.r.t. a given TBox $\mathcal{T}$.*

Proof. (Sketch) By Theorem 5.1 and Lemma 6.4, Algorithm 1″ is a decision procedure for the considered problem. It remains to show that the algorithm runs in exponential time. Let n be the size of $\mathcal{A} \cup \mathcal{T}$. As stated in the proof of [21, Lemma 6.4], each path of complex nodes constructed by Step 1 of Algorithm 1″ has length of rank $O(n^2)$. Analogously to the proofs of [21, Lemmas 6.2 and 6.4], it can be shown that Steps 2 and 3 of Algorithm 1″ are executed in $2^{O(n)}$ steps. Hence the complexity of Algorithm 1″ is of rank $2^{O(n^2)} \times 2^{O(n)}$, which is $2^{O(n^2)}$. ◁

6.3 On the Instance Checking Problem

Observe that $(\mathcal{A}, \mathcal{T}) \models \varphi(a)$ iff the ABox $\mathcal{A} \cup \{a : \overline{\varphi}\}$ is unsatisfiable w.r.t. $\mathcal{T}$. So, the instance checking problem is reduced to the problem of checking unsatisfiability of an ABox w.r.t. a TBox. What we are interested in is the *data complexity* of the instance checking problem, which is measured in the size of $\mathcal{A}$ when assuming that $\mathcal{A}$ is extensionally reduced and $\mathcal{T}$, φ, a are fixed. Here, $\mathcal{T}$, φ and a form a fixed query, while $\mathcal{A}$ varies as input data.

Theorem 6.6. *The data complexity of the instance checking problem in PDL is coNP-complete.*

Proof. Let $\mathcal{A}$ be an extensionally reduced ABox, $\mathcal{T}$ be a TBox, φ be a (traditional) formula in NNF, and a be a state variable. Consider the problem of checking whether $(\mathcal{A}, \mathcal{T}) \models \varphi(a)$.

Let p be a fresh proposition (not occurring in $\mathcal{A}$, $\mathcal{T}$, φ) and let $\mathcal{T}' = \mathcal{T} \cup \{\neg p \vee \varphi,\ p \vee \overline{\varphi}\}$ and $\mathcal{A}' = \mathcal{A} \cup \{a : \neg p\}$.

Observe that $\mathcal{T}'$ extends $\mathcal{T}$ with the formulas stating that p is equivalent to φ, and that $(\mathcal{A}, \mathcal{T}) \models \varphi(a)$ iff the ABox $\mathcal{A}'$ is unsatisfiable w.r.t. the TBox $\mathcal{T}'$.

Let n be the size of $\mathcal{A}$. The size of $\mathcal{A}' \cup \mathcal{T}'$ is thus of rank $O(n)$.

Consider an execution of Algorithm 1″ for the pair $\mathcal{A}'$ and $\mathcal{T}'$. As stated in the proof of [21, Lemma 6.4], each path of complex nodes constructed by Step 1 of Algorithm 1″ has length of rank $O(n^2)$. The sets $\mathcal{L}(v)$ and $rfs(v)$ of each complex node contain at most $O(n^2)$ formulas. Hence a nondeterministic execution of Step 1 of Algorithm 1″ runs in time $O(n^2) \times O(n^2)$. Since A' is extensionally reduced, The sets $\mathcal{L}(v)$ and $rfs(v)$ of each simple node v depends only on $\mathcal{T}'$. Since $\mathcal{T}'$ is fixed, Steps 2 and 3 of Algorithm 1″ are executed in time of rank $O(n^2)$. Hence the execution of Algorithm 1″ for $\mathcal{A}'$ and $\mathcal{T}'$ runs nondeterministically in polynomial time the size of $\mathcal{A}$, and therefore the instance checking problem $(\mathcal{A}, \mathcal{T}) \models \varphi(a)$ is in coNP.

The coNP-hardness follows from the fact that the instance checking problem in the description logic $\mathcal{ALC}$ is coNP-hard (see [25]). ◁

7 Conclusions

In this paper we first provided a tableau-based algorithm for checking satisfiability of a set of formulas in PDL. We then gave an ExpTime tableau decision procedure for checking consistency of an ABox w.r.t. a TBox in PDL ($\mathcal{ALC}_{reg}$).

Our latter procedure is the first optimal (ExpTime) tableau decision procedure not based on transformation for checking consistency of an ABox w.r.t. a TBox in PDL. Recall that, in [9]

the ABox is encoded by nominals, while in [10] the ABox is encoded by a concept assertion plus terminology axioms. Note that the approach based on transformation is not efficient in practice: in the well-known tutorial "Description Logics - Basics, Applications, and More", Horrocks and Sattler wrote *"direct algorithm/implementation instead of encodings"* and *"even simple domain encoding is disastrous with large numbers of roles"*.

The result that the data complexity of the instance checking problem in PDL is coNP-complete is first established in our paper.

Combining global caching for nodes representing objects not occurring as individuals in the ABox with backtracking for nodes representing individuals occurring in the ABox to obtain another ExpTime decision procedure is first studied by us in this paper.

Despite that our decision procedure for the case without ABoxes is based on Pratt's algorithm for PDL, our formulation of the tableau calculus for the procedure and our proof of its completeness are completely different than the ones of Pratt. Our decision procedure is formulated in a much simpler way. Note that Pratt's algorithm has been considered complicated: Donini and Massacci wrote in their paper [7] on ExpTime tableaux for $\mathcal{ALC}$ that they had proposed "the first simple tableau based decision procedure working in single exponential time" (here, note that $\mathcal{ALC}$ is a sub-logic of PDL), which in turn is considered by Baader and Sattler [3] still as "rather complicated". Also note that nobody has implemented Pratt's algorithm (except Pratt himself in the 70s, but his prototype is not available) and it is natural to ask why that algorithm, known since the 70s, remains unimplemented.

The idea of global caching comes from Pratt's paper on PDL, but it was discussed rather informally. Donini and Massacci in the mentioned paper on $\mathcal{ALC}$ stated that the caching optimization technique *"prunes heavily the search space but its unrestricted usage may lead to unsoundness [37]. It is conjectured that 'caching' leads to* ExpTime*-bounds but this has not been formally proved so far, nor the correctness of caching has been shown."*. Goré and Nguyen have recently formalized sound global caching [11, 12] for tableaux in a number of modal logics without the $*$ operator. Extending sound global caching for PDL would better be "formally proved" as done in our paper. Our extension for PDL considerably differs from [11, 12] :

- Due to the $*$ operator we have to check not only local consistency but also global consistency of the constructed "and-or" graph.
- We defined tableaux directly as "and-or" graphs with global caching, while in [11, 12] Goré and Nguyen used (traditional) tree-like tableaux and formulated global caching separately. Consequently, we do not have to prove soundness of global caching when having soundness and completeness of the calculus, while Goré and Nguyen [11, 12] had to prove soundness of global caching separately after having completeness of their calculi.

Comparing this paper with the independent paper [13] of Goré and Widmann, note that:

- The paper [13] considers only the problem of checking satisfiability of a formula in PDL, while the current paper considers also the problem of checking consistency of an ABox w.r.t. a TBox and the instance checking problem.
- The decision procedures given in [13] and the current paper for checking satisfiability of a formula in PDL are considerably different:
  - We leave "on-the-fly" propagation of satisfiability and unsatisfiability as optimizations [21], while Goré and Widmann incorporated such a propagation into the formal formulation of their algorithm. In our opinion, their formalization of that kind of propagation is quite inflexible: it requires depth-first-search and assumes that all children of a node must have been processed before computing propagation (of certain things) for that node.
  - Fulfilling eventualities is done substantially differently in [13] and the current paper.
- Proofs of correctness of the decision procedure given in [13] are not available to the public at the moment of writing the current paper.

We have applied our methods also to CPDL and regular grammar logics with converse (REG^c) to obtain complexity-optimal tableau decision procedures for these logics [19, 22]. In [20] we proved that the data complexity of the instance checking problem in REG^c is also coNP-complete. Such a

complexity can be shown also for CPDL and regular grammar logics without converse. As future work, we will extend the tableau prover TGC to deal also with PDL, CDPL and regular grammar logics with/without converse.

References

1. P. Abate, R. Goré, and F. Widmann. An on-the-fly tableau-based decision procedure for PDL. To appear in Proc. Methods for Modalities 2007.
2. F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of IJCAI'91*, pages 446–451, 1991.
3. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
4. D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In *Handbook of Automated Reasoning*, pages 1581–1634. Elsevier, 2001.
5. G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, 117-137:87–138, 2000.
6. P. Doherty, B. Dunin-Kęplicz, and A. Szałas. Dynamics of approximate information fusion. In M. Kryszkiewicz, J. Peters, H. Rybinski, and A. Skowron, editors, *Proc. RSEISP 2007*, number 4585 in LNAI, pages 668–677. Springer-Verlag, 2007.
7. F. Donini and F. Massacci. EXPTime tableaux for $\mathcal{ALC}$. *Art. Intelligence*, 124:87–138, 2000.
8. M.J. Fischer and R.E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.
9. G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Universita' di Roma "La Sapienza", 1995.
10. G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proceedings of KR'1996*, pages 316–327, 1996.
11. R. Goré and L.A. Nguyen. EXPTIME tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In N. Olivetti, editor, *Proc. of TABLEAUX 2007, LNAI 4548*, pages 133–148. Springer-Verlag, 2007.
12. R. Goré and L.A. Nguyen. Sound global caching for abstract modal tableaux. In H.-D. Burkhard et al, editor, *Proceedings of CS&P'2008*, pages 157–167, 2008.
13. R. Goré and F. Widmann. An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In R.A. Schmidt, editor, *Proceedings of CADE-22, LNAI 5663*, pages 437–452. Springer-Verlag, 2009.
14. J.Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
15. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
16. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proceedings of IJCAI-05*, pages 466–471, 2005.
17. `http://www.irit.fr/Lotrec/`.
18. L.A. Nguyen. An efficient tableau prover using global caching for the description logic ALC. *Fundamenta Informaticae*, 93(1-3):273–288, 2009.
19. L.A. Nguyen and A. Szałas. An optimal tableau decision procedure for converse-PDL. Accepted for KSE'2009.
20. L.A. Nguyen and A. Szałas. ExpTime tableau decision procedures for regular grammar logics with converse. Manuscript, available at `http://www.mimuw.edu.pl/~nguyen/creg-long.pdf`, 2009.
21. L.A. Nguyen and A. Szałas. Optimal tableau decision procedures for PDL. `http://arxiv.org/abs/0904.0721`, 2009.
22. L.A. Nguyen and A. Szałas. A tableau calculus for regular grammar logics with converse. In R.A. Schmidt, editor, *Proceedings of CADE-22, LNAI 5663*, pages 421–436. Springer-Verlag, 2009.
23. D. Niwiński and I. Walukiewicz. Games for the mu-calculus. *Theor. Comput. Sci.*, 163(1&2):99–116, 1996.
24. V.R. Pratt. A near-optimal method for reasoning about action. *J. Comput. Syst. Sci.*, 20(2):231–254, 1980.
25. A. Schaerf. Reasoning with individuals in concept languages. *Data Knowl. Eng.*, 13(2):141–176, 1994.
26. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of IJCAI'1991*, pages 466–471, 1991.
27. R.A. Schmidt. `http://www.cs.man.ac.uk/~schmidt/pdl-tableau/`.