Fundamenta Informaticae 143(1–2) (2016) 113–139 (last revised: March 22, 2018)
DOI 10.3233/FI-2016-1307
IOS Press

A Tractable Rule Language in the Modal and Description Logic That Combines CPDL with Regular Grammar Logic

Linh Anh Nguyen

Division of Knowledge and System Engineering for ICT
Faculty of Information Technology
Ton Duc Thang University
Ho Chi Minh City, Vietnam
nguyenanhlinh@tdt.edu.vn

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

Abstract. Combining CPDL (Propositional Dynamic Logic with Converse) and regular grammar logic results in an expressive modal logic denoted by CPDL$_{reg}$. This logic covers TEAMLOG, a logical formalism used to express properties of agents' cooperation in terms of beliefs, goals and intentions. It can also be used as a description logic for expressing terminological knowledge, in which both regular role inclusion axioms and CPDL-like role constructors are allowed. In this paper, we develop an expressive and tractable rule language called Horn-CPDL$_{reg}$. As a special property, this rule language allows the concept constructor "universal restriction" to appear on the left hand side of general concept inclusion axioms. We use a special semantics for Horn-CPDL$_{reg}$ that is based on pseudo-interpretations. It is called the constructive semantics and coincides with the traditional semantics when the concept constructor "universal restriction" is disallowed on the left hand side of concept inclusion axioms or when the language is used as an epistemic formalism and the accessibility relations are serial. We provide an algorithm with PTIME data complexity for checking whether a knowledge base in Horn-CPDL$_{reg}$ has a pseudo-model. This shows that the instance checking problem in Horn-CPDL$_{reg}$ with respect to the constructive semantics has PTIME data complexity.

Address for correspondence: Division of Knowledge and System Engineering for ICT, Faculty of Information Technology, Ton Duc Thang University, No. 19, Nguyen Huu Tho Street, Tan Phong Ward, District 7, Ho Chi Minh City, Vietnam

1. Introduction

Combining CPDL (Propositional Dynamic Logic with Converse) [1] and regular grammar logic [2, 3] results in an expressive modal logic denoted by CPDL$_{reg}$ [4]. This logic covers TEAMLOG [5, 6], a logical formalism used to express properties of agents' cooperation in terms of beliefs, goals and intentions. It can also be used as a description logic, in which both regular role inclusion axioms and CPDL-like role constructors are allowed.

Description logics (DLs) are variants of modal logics suitable for expressing terminological knowledge. They represent the domain of interest in terms of individuals, concepts and roles. A concept stands for a set of individuals, a role stands for a binary relation between individuals. In comparison with modal logic, concepts correspond to formulas, role names correspond to modal indices, roles correspond to programs in dynamic logic, and the concept constructors $\forall R.C$ and $\exists R.C$ correspond to the modalities $[R]C$ and $\langle R\rangle C$, respectively.

In this work, CPDL$_{reg}$ is considered as a DL and the objective is to develop an expressive rule language in CPDL$_{reg}$ that has PTIME data complexity.

1.1. Related Work and Motivation

The data complexity of the general Horn fragment in the basic DL $\mathcal{ALC}$ is NP-hard [7]. The hardness is caused by basic roles not being required to be serial (i.e., to satisfy the condition $\forall x \exists y\, R(x, y)$). A naive approach for overcoming the NP-hardness is to disallow the concept constructor $\forall R.C$ on the LHS (left hand side) of $\sqsubseteq$ in TBox axioms [8, 9, 10, 11, 12, 13, 14, 15, 16].

$\mathcal{EL}$ [9, 10], DL-Lite [17, 16], DLP [8], Horn-$\mathcal{SHIQ}$ [11] and Horn-$\mathcal{SROIQ}$ [15] are well-known rule languages in DLs with PTIME data complexity. The combined complexity of Horn fragments of DLs were considered, amongst others, in [18]. Some tractable Horn fragments of DLs without ABoxes have also been isolated in [9, 19]. To guarantee PTIME data or combined complexity, all of the rule languages in the mentioned works disallow the concept constructor $\forall R.C$ on the LHS of $\sqsubseteq$ in TBox axioms.

More sophisticated approaches for dealing with the mentioned NP-hardness are as follows:

- allowing $\forall R.C$ to appear on the LHS of $\sqsubseteq$ in TBox axioms when R is serial and using the traditional semantics for it,
- allowing a special kind of $\forall R.C$ like $\forall\exists R.C$ (defined as $\forall R.C \sqcap \exists R.C$) to appear on the LHS of $\sqsubseteq$ in TBox axioms and using the traditional semantics for it,
- allowing $\forall R.C$ to appear on the LHS of $\sqsubseteq$ in TBox axioms and using a special semantics for it.

As discussed below, our previous works on rule languages in propositional modal and description logics follow the first two of the above approaches.

In [20, 21] we studied Horn fragments of serial modal logics. They allow the constructor $[R]C$ to appear on the LHS of $\rightarrow$ in program clauses (like allowing $\forall R.C$ to appear on the LHS of $\sqsubseteq$ in TBox axioms) and have PTIME data complexity. The work [20] (resp. [21]) concerns constructing a logically smallest model of a positive logic program (resp. a Horn knowledge base) in serial regular grammar logic (resp. serial PDL).

The logic TEAMLOG [5, 6] is used to express properties of agents' cooperation in terms of beliefs, goals and intentions. In the joint work [22], we introduced a Horn fragment of TEAMLOG, called

Horn-TEAMLOG, and proved that it has PTIME data complexity. TEAMLOG can be translated into CPDL$_{reg}$ [4]. It is nearly serial in the sense that only the accessibility relations for goals are non-serial and the axioms about goals are simpler than the axioms about beliefs and intentions. Universal modal operators for beliefs, common beliefs, intentions and mutual intentions are allowed on the LHS of $\rightarrow$ in Horn-TEAMLOG program clauses. For goals of an agent σ, the combined modal operator $[G_\sigma]_\diamond\varphi$ defined as $[G_\sigma]\varphi \wedge \langle G_\sigma\rangle\varphi$ is allowed instead of the universal modal operator $[G_\sigma]\varphi$ on the LHS of $\rightarrow$ in Horn-TEAMLOG program clauses.

In [23] we introduced the deterministic Horn fragment of Test-free PDL, which is not serial. The fragment allows the modal operator $[\pi]_\diamond\varphi$ to appears on the LHS of $\rightarrow$ in program clauses. This modal operator is stronger than $[\pi]\varphi \wedge \langle\pi\rangle\varphi$. The formula $[\pi]_\diamond\varphi$ means that $[\pi]\varphi$ holds and every partial run of π (starting from the considered state in the considered Kripke model) is not blocked. The deterministic Horn fragment of Test-free PDL has PTIME data complexity.

In the context of DLs, the first rule language with PTIME data complexity that allows a form of $\forall R.C$ on the LHS of $\sqsubseteq$ in TBox axioms was introduced by us in [24]. It is the deterministic Horn fragment of $\mathcal{ALC}$, for which the constructor $\forall\exists R.C$, defined as $\forall R.C \sqcap \exists R.C$ [19], is allowed on the LHS of $\sqsubseteq$ in TBox axioms. We estimated the data complexity of that fragment by providing a bottom-up method for constructing a logically smallest pseudo-model for a given deterministic positive knowledge base in the restricted language. In [7] we applied the method of [24] to a Horn fragment of the regular DL $\mathcal{R}eg$, which we denote by Horn-$\mathcal{R}eg$. The works [24, 7] use the bottom-up approach and the traditional semantics for $\forall\exists R.C$. However, the techniques of [24, 7] are already related to a special notion called pseudo-interpretation as in [23], which is on the way towards untraditional semantics.

In the joint work [25], we introduced a Horn description logic called Horn-DL, which is strictly and essentially richer than Horn-$\mathcal{R}eg$, Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SROIQ}$, while still having PTIME data complexity. In comparison with Horn-$\mathcal{R}eg$ [7], Horn-DL additionally allows inverse roles, nominals, qualified number restrictions, the $\exists r.\mathsf{Self}$ constructor, the universal role as well as assertions of the form $\mathsf{disjoint}(s, s')$, $\mathsf{irreflexive}(s)$, $\neg s(a, b)$, $a \not\doteq b$. In comparison with Horn-$\mathcal{SROIQ}$, Horn-DL additionally allows the universal role and assertions of the form $\mathsf{irreflexive}(s)$, $\neg s(a, b)$, $a \not\doteq b$. More importantly, in contrast to all the well-known Horn fragments $\mathcal{EL}$ [9, 10], DL-Lite [17], DLP [8], Horn-$\mathcal{SHIQ}$ [11], Horn-$\mathcal{SROIQ}$ [15] of DLs, Horn-DL allows the concept constructor $\forall\exists R.C$ to appear on the LHS of $\sqsubseteq$ in TBox axioms.

The objective of this paper is to formulate an as rich as possible Horn fragment in CPDL$_{reg}$ together with an appropriate semantics for it. As discussed in [7, 25], for $R \in \mathbf{R}$, the concept constructor $\forall\exists R.C$ is more constructive than $\forall R.C$ on the LHS of $\sqsubseteq$ in TBox axioms. For the case when R is not a basic role, a constructor similar to $[\pi]_\diamond\varphi$ of [23] seems to be too strong and complicated for practical applications. A natural question is: *Can the concept constructor $\forall R.C$ be directly used on the LHS of $\sqsubseteq$ in TBox axioms?* Our answer is: *Yes, why not? To obtain the* PTIME *data complexity, just formulate and use an appropriate semantics for that constructor.*

1.2. Our Contributions

We introduce a tractable rule language called Horn-CPDL$_{reg}$, which is a fragment of CPDL$_{reg}$. As a special property, it allows the concept constructors $\forall\exists R.C$ and $\forall R.C$ to appear on the LHS of $\sqsubseteq$ in TBox axioms. We use a special semantics for Horn-CPDL$_{reg}$ that is based on pseudo-interpretations. It is called the *constructive semantics* and coincides with the traditional semantics when the concept

constructor $\forall R.C$ is disallowed on the LHS of TBox axioms or when the language is used as an epistemic formalism and the accessibility relations are serial. We provide an algorithm with PTIME data complexity for checking whether a knowledge base in Horn-CPDL$_{reg}$ has a pseudo-model. This shows that the instance checking problem in Horn-CPDL$_{reg}$ with respect to the constructive semantics has PTIME data complexity.

1.3. The Structure of This Paper

The rest of this paper is structured as follows. Section 2 recalls the notation and semantics of CPDL$_{reg}$. Section 3 defines the rule language Horn-CPDL$_{reg}$. Section 4 presents the constructive semantics of Horn-CPDL$_{reg}$ and its properties. Section 5 provides our algorithm for checking whether a given knowledge base in Horn-CPDL$_{reg}$ has a pseudo-model. Section 6 contains concluding remarks.

2. Preliminaries

Our language uses a finite set $\mathbf{C}$ of *concept names*, a finite set $\mathbf{R}_+$ of *role names*, and a finite set $\mathbf{I}$ of *individual names*. We use letters like a, b to denote individual names, letters like A, B to denote concept names, and letters like r, s to denote role names. We use $\overline{r}$ to denote the *inverse* of r. For $R = \overline{r}$, let $\overline{R}$ stand for r. Let $\mathbf{R}_- = \{\overline{r} \mid r \in \mathbf{R}_+\}$ and $\mathbf{R} = \mathbf{R}_+ \cup \mathbf{R}_-$. We call the roles from $\mathbf{R}$ *basic roles*.

2.1. Regular RBoxes

A *context-free semi-Thue system* $\mathcal{S}$ over $\mathbf{R}$ is a finite set of context-free production rules $R \to S_1 \dots S_k$ over alphabet $\mathbf{R}$ (i.e., $R, S_1, \dots, S_k \in \mathbf{R}$). It is *symmetric* if, for every rule $R \to S_1 \dots S_k$ of $\mathcal{S}$, the rule $\overline{R} \to \overline{S}_k \dots \overline{S}_1$ is also in $\mathcal{S}$.[1] It is *regular* if, for every $R \in \mathbf{R}$, the set of words derivable from R using the system is a regular language over $\mathbf{R}$.

A context-free semi-Thue system is like a context-free grammar, but it has no designated start symbol and there is no distinction between terminal and non-terminal symbols. We assume that, for $R \in \mathbf{R}$, the word R is derivable from R using such a system.

A *role inclusion axiom* (RIA for short) is an expression of the form $S_1 \circ \dots \circ S_k \sqsubseteq R$, where $k \geq 0$ and $S_1, \dots, S_k, R \in \mathbf{R}$. In the case $k = 0$, the LHS of the inclusion axiom stands for the empty word ε.

A *regular RBox* $\mathcal{R}$ is a finite set of RIAs such that

$$\{R \to S_1 \dots S_k \mid (S_1 \circ \dots \circ S_k \sqsubseteq R) \in \mathcal{R}\}$$

is a symmetric regular semi-Thue system $\mathcal{S}$ over $\mathbf{R}$. We assume that $\mathcal{R}$ is given together with a mapping $\mathbf{A}$ that associates every $R \in \mathbf{R}$ with a finite automaton $\mathbf{A}_R$ recognizing the words derivable from R using $\mathcal{S}$. We call $\mathbf{A}$ the *RIA-automaton-specification* of $\mathcal{R}$.

Recall that a *finite automaton* A over alphabet $\mathbf{R}$ is a tuple $\langle \mathbf{R}, Q, q_0, \delta, F \rangle$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta \subseteq Q \times \mathbf{R} \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. A *run* of A on a word $R_1 \dots R_k$ over alphabet $\mathbf{R}$ is a finite sequence of states $q_0, q_1, \dots, q_k$ such that $\delta(q_{i-1}, R_i, q_i)$ 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 all words accepted by A is denoted by $\mathcal{L}(\mathsf{A})$.

[1] In the case $k = 0$, the right hand sides of the rules stand for ε.

Example 2.1. Let $\mathbf{R}_+ = \{r\}$ and $\mathcal{R} = \{\overline{r} \circ r \sqsubseteq r,\ r \circ \overline{r} \sqsubseteq \overline{r}\}$. The symmetric regular semi-Thue system corresponding to $\mathcal{R}$ is $\mathcal{S} = \{r \to \overline{r}r,\ \overline{r} \to \overline{r}r\}$. The set of words derivable from r (resp. $\overline{r}$) using $\mathcal{S}$ is a regular language characterized by the regular expression $r \cup (\overline{r}; (\overline{r} \cup r)^*; r)$ (resp. $\overline{r} \cup (\overline{r}; (\overline{r} \cup r)^*; r)$). Hence, $\mathcal{R}$ is a regular RBox, whose RIA-automaton-specification $\mathbf{A}$ is specified by:

$$\begin{aligned} \mathbf{A}_r &= \langle \mathbf{R}, \{0,1,2\}, 0, \{\langle 0,r,1\rangle, \langle 0,\overline{r},2\rangle, \langle 2,r,2\rangle, \langle 2,\overline{r},2\rangle, \langle 2,r,1\rangle\}, \{1\}\rangle \\ \mathbf{A}_{\overline{r}} &= \langle \mathbf{R}, \{0,1,2\}, 0, \{\langle 0,\overline{r},1\rangle, \langle 0,\overline{r},2\rangle, \langle 2,r,2\rangle, \langle 2,\overline{r},2\rangle, \langle 2,r,1\rangle\}, \{1\}\rangle. \end{aligned}$$

□

See [25] for other examples of RBoxes and RIA-automaton-specifications.

Observe that every regular set of RIAs in $\mathcal{SROIQ}$ [26] and Horn-$\mathcal{SROIQ}$ [15] is a regular RBox by our definition. However, the above RBox $\mathcal{R}$ shows that the converse does not hold. Roughly speaking, using the notion of regular expressions, "regularity" of a set of RIAs in $\mathcal{SROIQ}$ [26] and Horn-$\mathcal{SROIQ}$ [15] allows only a bounded nesting depth of the star operator *, while "regularity" of a regular RBox in Horn-$\mathcal{R}eg^I$ is not so restricted. That is, our notion of regular RBox is more general than the notion of regular set of RIAs in $\mathcal{SROIQ}$ [26] and Horn-$\mathcal{SROIQ}$ [15].

2.2. Notation and Semantics of CPDL$_{reg}$

Concepts and *roles* are defined, respectively, by the following BNF grammar rules, where $A \in \mathbf{C}$ and $r \in \mathbf{R}_+$:

$$\begin{aligned} C &::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C \\ R &::= \varepsilon \mid r \mid \overline{R} \mid R \circ R \mid R \sqcup R \mid R^* \mid C? \end{aligned}$$

We use letters like C, D to denote concepts, and letters like R, S to denote roles.

A *terminological axiom*, also called a *TBox axiom*, is an expression of the form $C \sqsubseteq D$. A *TBox* is a finite set of *TBox axioms*. An *ABox* is a finite set of assertions of the form $C(a)$ or $r(a,b)$. A *knowledge base* in CPDL$_{reg}$ is a tuple $\langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ consisting of a regular RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set called the *domain* of $\mathcal{I}$ and $\cdot^{\mathcal{I}}$ is a mapping called the *interpretation function* of $\mathcal{I}$ that associates each individual name $a \in \mathbf{I}$ with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each concept name $A \in \mathbf{C}$ with a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each role name $r \in \mathbf{R}_+$ with a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is extended to complex concepts and complex roles as shown in Figure 1.

Given an interpretation $\mathcal{I}$ and an axiom/assertion φ, the satisfaction relation $\mathcal{I} \models \varphi$ is defined as follows, where $\circ$ on the right hand side of "if" stands for the composition of binary relations:

$$\begin{aligned} &\mathcal{I} \models S_1 \circ \cdots \circ S_k \sqsubseteq R && \text{if} && S_1^{\mathcal{I}} \circ \cdots \circ S_k^{\mathcal{I}} \subseteq R^{\mathcal{I}} \\ &\mathcal{I} \models \varepsilon \sqsubseteq R && \text{if} && \varepsilon^{\mathcal{I}} \sqsubseteq R^{\mathcal{I}} \\ &\mathcal{I} \models C \sqsubseteq D && \text{if} && C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \\ &\mathcal{I} \models C(a) && \text{if} && a^{\mathcal{I}} \in C^{\mathcal{I}} \\ &\mathcal{I} \models r(a,b) && \text{if} && \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in r^{\mathcal{I}}. \end{aligned}$$

If $\mathcal{I} \models \varphi$, then we say that $\mathcal{I}$ *validates* φ.

$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset$	$\varepsilon^{\mathcal{I}} = \{\langle x, x\rangle \mid x \in \Delta^{\mathcal{I}}\}$
$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$\overline{R}^{\mathcal{I}} = (R^{\mathcal{I}})^{-1}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$(R \circ S)^{\mathcal{I}} = R^{\mathcal{I}} \circ S^{\mathcal{I}}$
$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$(R \sqcup S)^{\mathcal{I}} = R^{\mathcal{I}} \cup S^{\mathcal{I}}$
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y\, (\langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}})\}$	$(R^*)^{\mathcal{I}} = (R^{\mathcal{I}})^*$
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y\, (\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$	$(C?)^{\mathcal{I}} = \{\langle x, x\rangle \mid x \in C^{\mathcal{I}}\}$

Figure 1. Interpretation of complex concepts and complex roles.

An interpretation $\mathcal{I}$ is a *model* of an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ or an ABox $\mathcal{A}$ if it validates all the axioms/assertions of that "box". It is a *model* of a knowledge base $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$, denoted by $\mathcal{I} \models KB$, if it is a model of $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}$.

A knowledge base is *satisfiable* if it has a model. For a knowledge base KB, we write $KB \models \varphi$ to mean that every model of KB validates φ. If $KB \models C(a)$, then we say that a is an *instance* of C w.r.t. KB.

Example 2.2. Let $\mathbf{I} = \{Lily, Jack\}$, $\mathbf{R}_+ = \{hasSon, hasDaughter\}$ and $\mathbf{C} = \{Male, Female, A, B, C, D, E\}$. One can imagine A, B, C, D, E as some properties about genes or diseases. Let $\mathcal{R} = \emptyset$, $\mathcal{A} = \{A(Lily), hasSon(Lily, Jack), (\exists hasSon.\top)(Jack)\}$ and let $\mathcal{T}$ be the TBox consisting of the following axioms (where $\sqcap$ binds more weakly than $\forall$):

$$\top \sqsubseteq \forall hasSon.Male, \tag{1}$$
$$\top \sqsubseteq \forall hasDaughter.Female, \tag{2}$$
$$Male \sqcap Female \sqsubseteq \bot, \tag{3}$$
$$A \sqsubseteq \forall hasSon.B \sqcap \forall hasDaughter.C \sqcap D, \tag{4}$$
$$Male \sqcap B \sqsubseteq \forall (hasSon \sqcup hasDaughter)^*.D, \tag{5}$$
$$Female \sqcap C \sqsubseteq \forall (hasSon \sqcup hasDaughter)^*.D, \tag{6}$$
$$\forall (hasSon \sqcup hasDaughter)^*.D \sqsubseteq E. \tag{7}$$

Let $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ and consider the interpretation $\mathcal{I}$ specified by:

- $\Delta^{\mathcal{I}} = \{a, b, c\}$, $Lily^{\mathcal{I}} = a$, $Jack^{\mathcal{I}} = b$,
- $hasSon^{\mathcal{I}} = \{\langle a, b\rangle, \langle b, c\rangle\}$, $hasDaughter^{\mathcal{I}} = \emptyset$,
- $Female^{\mathcal{I}} = \{a\}$, $Male^{\mathcal{I}} = \{b, c\}$,
- $A^{\mathcal{I}} = \{a\}$, $B^{\mathcal{I}} = \{b\}$, $C^{\mathcal{I}} = \emptyset$, $D^{\mathcal{I}} = E^{\mathcal{I}} = \{a, b, c\}$.

It is easy to see that $\mathcal{I}$ is a model of KB. Note, however, that KB has also other models and $\mathcal{I}$ is not the "least" model of KB (e.g., $\mathcal{I} \models (\forall hasDaughter.Male)(Lily)$ but $KB \not\models (\forall hasDaughter.Male)(Lily)$). Also observe that $KB \models E(Lily)$. □

The *size* of a concept, a role, an assertion or an axiom φ is defined to be the number of bits needed to encode φ in the usual way. The *size* of an ABox is the sum of the sizes of its assertions. The *size* of a TBox is the sum of the sizes of its axioms.

A *reduced ABox* is a finite set of assertions of the form $A(a)$, $\neg A(a)$ or $r(a, b)$. The *data complexity* of the instance checking problem $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle \models C(a)$ is defined when $\mathcal{A}$ is a reduced ABox and is measured w.r.t. the size of $\mathcal{A}$, while assuming that $\mathbf{R}_+$, $\mathcal{R}$, $\mathcal{T}$ and $C(a)$ are fixed.

3. The Horn-CPDL$_{reg}$ Fragment

A *Horn-CPDL$_{reg}$ TBox axiom* is an expression of the form $C_l \sqsubseteq C_r$, where l stands for "left", r stands for "right", C_l and C_r are concepts defined by the following BNF grammar:

$$C_l ::= \top \mid A \mid C_l \sqcap C_l \mid C_l \sqcup C_l \mid \exists R_{l\exists}.C_l \mid \forall R_{l\forall}.C_l \mid \forall\exists r.C_l \mid \forall\exists \overline{r}.C_l \quad (8)$$

$$R_{l\exists} ::= r \mid \overline{R_{l\exists}} \mid R_{l\exists} \circ R_{l\exists} \mid R_{l\exists} \sqcup R_{l\exists} \mid R^*_{l\exists} \mid C_l? \quad (9)$$

$$R_{l\forall} ::= r \mid \overline{R_{l\forall}} \mid R_{l\forall} \circ R_{l\forall} \mid R_{l\forall} \sqcup R_{l\forall} \mid R^*_{l\forall} \mid (\neg C_l)? \quad (10)$$

$$C_r ::= \top \mid \bot \mid A \mid \neg C_l \mid C_r \sqcap C_r \mid \neg C_l \sqcup C_r \mid \exists R_{r\exists}.C_r \mid \forall R_{l\exists}.C_r \quad (11)$$

$$R_{r\exists} ::= r \mid \overline{R_{r\exists}} \mid R_{r\exists} \circ R_{r\exists} \mid C_r? \quad (12)$$

A *Horn-CPDL$_{reg}$ TBox* is a finite set of *Horn-CPDL$_{reg}$ TBox axioms*.
For example, the TBox given in Example 2.2 is a Horn-CPDL$_{reg}$ TBox.
A *Horn-CPDL$_{reg}$ clause* is a TBox axiom of the form $C_1 \sqcap \ldots \sqcap C_k \sqsubseteq D$ or $\top \sqsubseteq D$, where:

- each C_i is of the form A, $\exists R_{l\exists}.A$, $\forall R_{l\forall}.A$, $\forall\exists r.A$ or $\forall\exists \overline{r}.A$,
- D is of the form[2] $\bot$, $\exists r.\top$, $\exists \overline{r}.\top$, A, $\exists r.A$, $\exists \overline{r}.A$ or $\forall R_{l\exists}.A$,
- $R_{l\exists}$ and $R_{l\forall}$ are now restricted by the following BNF grammar:

$$R_{l\exists} ::= r \mid \overline{r} \mid R_{l\exists} \circ R_{l\exists} \mid R_{l\exists} \sqcup R_{l\exists} \mid R^*_{l\exists} \mid A? \quad (13)$$

$$R_{l\forall} ::= r \mid \overline{r} \mid R_{l\forall} \circ R_{l\forall} \mid R_{l\forall} \sqcup R_{l\forall} \mid R^*_{l\forall} \mid (\neg A)? \quad (14)$$

A *clausal Horn-CPDL$_{reg}$ TBox* consists of Horn-CPDL$_{reg}$ clauses.
A *Horn-CPDL$_{reg}$ ABox* is a finite set of assertions of the form $C_r(a)$ or $r(a, b)$, where C_r is a concept of the form specified by (11).
A *Horn-CPDL$_{reg}$ knowledge base* is a tuple $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ consisting of a regular RBox $\mathcal{R}$, a Horn-CPDL$_{reg}$ TBox $\mathcal{T}$ and a Horn-CPDL$_{reg}$ ABox $\mathcal{A}$. When $\mathcal{T}$ is a clausal Horn-CPDL$_{reg}$ TBox and $\mathcal{A}$ is a reduced ABox, we call such a knowledge base a *clausal Horn-CPDL$_{reg}$ knowledge base*.
A *Horn-CPDL$_{reg}$ query* for the instance checking problem is an expression of the form $C(a)$, where $a \in \mathbf{I}$ and C is a concept of the family C_l specified by (8).
Note: When considering Horn-CPDL$_{reg}$, we allow only knowledge bases and queries in this language. Thus, a considered RBox must be regular and given together with its RIA-automaton-specification. Also recall that the data complexity is measured under the assumption that the considered ABox is reduced.

[2] A clause of the form $C \sqsubseteq \exists R.\top$ can be replaced by $C \sqsubseteq \exists R.A_\top$, where $A_\top$ is a fresh concept name. We allow such clauses just for convenience.

$\bot^{\mathcal{I}} = \emptyset$	$\varepsilon^{\mathcal{I}_\exists} = \{\langle x, x\rangle \mid x \in \Delta^{\mathcal{I}}\}$	$\varepsilon^{\mathcal{I}_\forall} = \varepsilon^{\mathcal{I}_\exists}$
$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$	$\overline{R}^{\mathcal{I}_\exists} = (R^{\mathcal{I}_\exists})^{-1}$	$\overline{R}^{\mathcal{I}_\forall} = (R^{\mathcal{I}_\forall})^{-1}$
$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$(R \circ S)^{\mathcal{I}_\exists} = R^{\mathcal{I}_\exists} \circ S^{\mathcal{I}_\exists}$	$(R \circ S)^{\mathcal{I}_\forall} = R^{\mathcal{I}_\forall} \circ S^{\mathcal{I}_\forall}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$(R \sqcup S)^{\mathcal{I}_\exists} = R^{\mathcal{I}_\exists} \cup S^{\mathcal{I}_\exists}$	$(R \sqcup S)^{\mathcal{I}_\forall} = R^{\mathcal{I}_\forall} \cup S^{\mathcal{I}_\forall}$
$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$(R^*)^{\mathcal{I}_\exists} = (R^{\mathcal{I}_\exists})^*$	$(R^*)^{\mathcal{I}_\forall} = (R^{\mathcal{I}_\forall})^*$
$(\forall\exists R.C)^{\mathcal{I}} = (\forall R.C \sqcap \exists R.C)^{\mathcal{I}}$	$(C?)^{\mathcal{I}_\exists} = \{\langle x, x\rangle \mid C^{\mathcal{I}}(x)\}$	$(C?)^{\mathcal{I}_\forall} = (C?)^{\mathcal{I}_\exists}$
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y\,(\langle x, y\rangle \in R^{\mathcal{I}_\forall} \Rightarrow y \in C^{\mathcal{I}})\}$		
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y\,(\langle x, y\rangle \in R^{\mathcal{I}_\exists} \wedge y \in C^{\mathcal{I}})\}$		

Figure 2. The meaning of complex concepts and complex roles in a pseudo-interpretation.

Example 3.1. The knowledge base KB given in Example 2.2 is a Horn-CPDL$_{reg}$ knowledge base. It can be converted to a clausal Horn-CPDL$_{reg}$ knowledge base $KB_2 = \langle \mathcal{R}, \mathcal{T}_2, \mathcal{A}_2\rangle$ as follows:

- $\mathcal{A}_2$ is obtained from $\mathcal{A}$ by replacing the assertion $(\exists hasSon.\top)(Jack)$ by $F(Jack)$, where F is a new concept name (and now $\mathbf{C} = \{Male, Female, A, B, C, D, E, F\}$);
- $\mathcal{T}_2$ is obtained from $\mathcal{T}$ by adding the axiom $F \sqsubseteq \exists hasSon.\top$ and replacing the axiom (4) by $(A \sqsubseteq \forall hasSon.B)$, $(A \sqsubseteq \forall hasDaughter.C)$ and $(A \sqsubseteq D)$. □

4. The Constructive Semantics of Horn-CPDL$_{reg}$

The objective is to define an appropriate satisfaction relation $KB \models\!\!\!\sim C(a)$ for a Horn-CPDL$_{reg}$ knowledge base KB and a Horn-CPDL$_{reg}$ query $C(a)$. The intention is that $KB \models\!\!\!\sim C(a)$ iff $C(a)$ can be derived from KB by constructive rules as in intuitionistic logic. However, this is beyond the scope of the current paper. Here, we are satisfied with the following properties:

- If $KB \models\!\!\!\sim C(a)$, then $KB \models C(a)$. That is, $\models\!\!\!\sim$ is sound w.r.t. the traditional semantics.
- If KB and C are specified without using the constructor $\forall R_{l\forall}.C_l$ in the grammar rule (8), then $KB \models\!\!\!\sim C(a)$ iff $KB \models C(a)$.
- If $\{\top \sqsubseteq \exists R.\top \mid R \in \mathbf{R}\} \subseteq \mathcal{T}$ (i.e., all $R \in \mathbf{R}$ are serial), then $KB \models\!\!\!\sim C(a)$ iff $KB \models C(a)$.

4.1. Pseudo-Interpretations

Pseudo-interpretations were introduced by us in [24, 23, 7]. Here, we extend that notion for CPDL$_{reg}$ to deal with inverse roles, using a slightly different notation that is closer to the traditional notation of DLs.

Definition 4.1. A *pseudo-interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set called the *domain* of $\mathcal{I}$ and $\cdot^{\mathcal{I}}$ is a mapping called the *interpretation function* of $\mathcal{I}$ that associates each individual name $a \in \mathbf{I}$ with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each concept name $A \in \mathbf{C}$ with a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each role name $r \in \mathbf{R}_+$ with a pair $\langle r^{\mathcal{I}_\exists}, r^{\mathcal{I}_\forall}\rangle$ of binary relations such that:

- $r^{\mathcal{I}_\exists} \subseteq r^{\mathcal{I}_\forall} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$,
- for every $x \in \Delta^{\mathcal{I}}$, if $Y = \{y \mid \langle x, y\rangle \in r^{\mathcal{I}_\exists}\} \neq \emptyset$, then $\{y \mid \langle x, y\rangle \in r^{\mathcal{I}_\forall}\} = Y$.

The interpretation function $\cdot^{\mathcal{I}}$ is extended to complex concepts and complex roles as shown in Figure 2. □

Observe that, given a pseudo-interpretation $\mathcal{I}$ and a role R, we have that $R^{\mathcal{I}_\exists} \subseteq R^{\mathcal{I}_\forall}$, and $(\forall R.C)^{\mathcal{I}}$ may differ from $(\neg\exists R.\neg C)^{\mathcal{I}}$. If $\langle x, y\rangle \in R^{\mathcal{I}_\exists}$, then we call $\langle x, y\rangle$ a *firm* R-edge. If $\langle x, y\rangle \in R^{\mathcal{I}_\forall} \setminus R^{\mathcal{I}_\exists}$, then we call $\langle x, y\rangle$ a *pseudo* R-edge.

Definition 4.2. Given a pseudo-interpretation $\mathcal{I}$ and an axiom/assertion φ, the satisfaction relation $\mathcal{I} \models\!\!\!\sim \varphi$ is defined as follows:

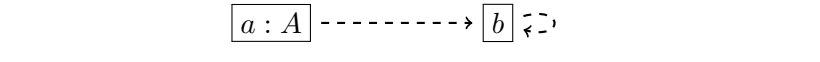
$$
\begin{array}{lll}
\mathcal{I} \models\!\!\!\sim S_1 \circ \cdots \circ S_k \sqsubseteq R & \text{if} & S_1^{\mathcal{I}_\exists} \circ \cdots \circ S_k^{\mathcal{I}_\exists} \subseteq R^{\mathcal{I}_\exists} \text{ and } S_1^{\mathcal{I}_\forall} \circ \cdots \circ S_k^{\mathcal{I}_\forall} \subseteq R^{\mathcal{I}_\forall} \\
\mathcal{I} \models\!\!\!\sim \varepsilon \sqsubseteq R & \text{if} & \varepsilon^{\mathcal{I}_\exists} \sqsubseteq R^{\mathcal{I}_\exists} \\
\mathcal{I} \models\!\!\!\sim C \sqsubseteq D & \text{if} & C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \\
\mathcal{I} \models\!\!\!\sim C(a) & \text{if} & a^{\mathcal{I}} \in C^{\mathcal{I}} \\
\mathcal{I} \models\!\!\!\sim r(a, b) & \text{if} & \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in r^{\mathcal{I}_\exists}.
\end{array}
$$

If $\mathcal{I} \models\!\!\!\sim \varphi$, then we say that $\mathcal{I}$ *validates* φ. A pseudo-interpretation $\mathcal{I}$ is a *pseudo-model* of an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ or an ABox $\mathcal{A}$ if it validates all the axioms/assertions of that "box". It is a *pseudo-model* of a knowledge base $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$, denoted by $\mathcal{I} \models\!\!\!\sim KB$, if it is a pseudo-model of $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}$. A knowledge base is *satisfiable w.r.t. the constructive semantics* if it has a pseudo-model. We define that $\langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle \models\!\!\!\sim C(a)$ if, for every pseudo-model $\mathcal{I}$ of $\langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$, it holds that $\mathcal{I} \models\!\!\!\sim C(a)$. □

Example 4.3. Let $\mathbf{I} = \{a\}$, $\mathbf{R}_+ = \{r\}$, $\mathbf{C} = \{A, B, C, D, E\}$, $\mathcal{R} = \emptyset$, $\mathcal{A} = \{A(a)\}$ and let $\mathcal{T}$ be the TBox consisting of the following axioms:

$$
\begin{array}{l}
A \sqcap \forall r.B \sqsubseteq C, \\
A \sqcap \exists r.\top \sqsubseteq D, \\
C \sqcup D \sqsubseteq E.
\end{array}
$$

Then $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ is a Horn-CPDL$_{reg}$ knowledge base. Observe that $KB \not\models C(a)$ and $KB \not\models D(a)$, but $KB \models E(a)$ although $C \sqcup D \sqsubseteq E$ is the only axiom of $\mathcal{T}$ that "defines" E. This partially explains the nondeterminism problem of allowing the constructor $\forall r.B$ on the LHS of $\sqsubseteq$ in TBox axioms when using the traditional semantics. Consider the pseudo-interpretation $\mathcal{I}$ illustrated and specified by:



$\Delta^{\mathcal{I}} = \{a, b\}$, $a^{\mathcal{I}} = a$, $A^{\mathcal{I}} = \{a\}$, $B^{\mathcal{I}} = C^{\mathcal{I}} = D^{\mathcal{I}} = E^{\mathcal{I}} = \emptyset$, $r^{\mathcal{I}_\exists} = \emptyset$, $r^{\mathcal{I}_\forall} = \{\langle a, b\rangle, \langle b, b\rangle\}$. Observe that $\mathcal{I} \models\!\!\!\sim KB$ and $\mathcal{I} \not\models E(a)$. This shows that the constructive semantics is "weaker" (i.e., allows to derive fewer logical consequences) than the traditional semantics. □

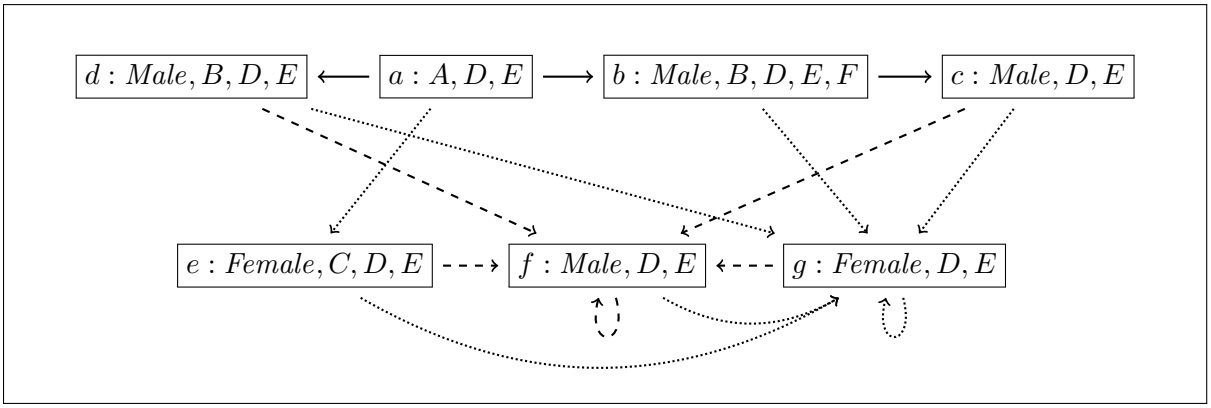


Figure 3. An illustration for Example 4.4.

Example 4.4. Reconsider the clausal Horn-CPDL$_{reg}$ knowledge base KB_2 specified in Example 3.1 and let $\mathcal{I}$ be the pseudo-interpretation illustrated by the graph in Figure 3, where:

- $\Delta^{\mathcal{I}} = \{a, \ldots, g\}$, $Lily^{\mathcal{I}} = a$ and $Jack^{\mathcal{I}} = b$,
- for $X \in \mathbf{C}$, $X^{\mathcal{I}}$ consists of the elements $y \in \Delta^{\mathcal{I}}$ such that X is displayed in the node y,
- $hasSon^{\mathcal{I}_\exists}$ consists of the pairs that are displayed by the normal edges,
- $hasSon^{\mathcal{I}_\forall}$ extends $hasSon^{\mathcal{I}_\exists}$ with the pairs that are displayed by the dashed edges,
- $hasDaughter^{\mathcal{I}_\exists} = \emptyset$, $hasDaughter^{\mathcal{I}_\forall}$ consists of the pairs that are displayed by the dotted edges.

Observer that $\mathcal{I} \models\!\!\!\sim KB_2$. Furthermore, it can be proved that $\mathcal{I}$ has the following special property: if φ is a Horn-CPDL$_{reg}$ query that does not use inverse roles, then $\mathcal{I} \models\!\!\!\sim \varphi$ iff $KB_2 \models\!\!\!\sim \varphi$ (that is, $\mathcal{I}$ is the "least" pseudo-model of KB_2 w.r.t. Horn-CPDL$_{reg}$ queries that do not use inverse roles). □

Remark 4.5. An interpretation $\mathcal{I}$ can be treated as a pseudo-interpretation with $r^{\mathcal{I}_\exists} = r^{\mathcal{I}_\forall} = r^{\mathcal{I}}$ for all $r \in \mathbf{R}_+$. Thus, given an interpretation $\mathcal{I}$, $\mathcal{I} \models KB$ iff $\mathcal{I} \models\!\!\!\sim KB$, and $\mathcal{I} \models C(a)$ iff $\mathcal{I} \models\!\!\!\sim C(a)$. Conversely, a pseudo-interpretation $\mathcal{I}$ satisfying $r^{\mathcal{I}_\exists} = r^{\mathcal{I}_\forall} = r^{\mathcal{I}}$ for all $r \in \mathbf{R}_+$ can be treated as an interpretation. In particular, if $\mathcal{I} \models (\top \sqsubseteq \exists R.\top)$ for all $R \in \mathbf{R}$, then $\mathcal{I}$ can be treated as an interpretation.

4.2. Conversion to the Clausal Form

A role is in the *inverse normal form* if in its construction the inverse operation is applied only to role names. Every role R can be transformed to an equivalent role S in the inverse normal form (such that $R^{\mathcal{I}} = S^{\mathcal{I}}$ in every interpretation $\mathcal{I}$). For example, $\overline{(r_1 \sqcup r_2) \circ r_3^*} \equiv \overline{r_3}^* \circ (\overline{r_1} \sqcup \overline{r_2})$.

Proposition 4.6. Let $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ be a Horn-CPDL$_{reg}$ knowledge base.

1. If $C(a)$ is a Horn-CPDL$_{reg}$ query, then, for the Horn-CPDL$_{reg}$ knowledge base $KB' = \langle \mathcal{R}, \mathcal{T} \cup \{C \sqsubseteq A\}, \mathcal{A} \cup \{\neg A(a)\} \rangle$, where A is a fresh concept name, we have that:
   (a) $KB \models\!\!\!\sim C(a)$ iff KB' does not have any pseudo-model,
   (b) $KB \models C(a)$ iff KB' does not have any model.

2. KB can be converted in polynomial time in the sizes of $\mathcal{T}$ and $\mathcal{A}$ to a Horn-CPDL$_{reg}$ knowledge base $KB' = \langle \mathcal{R}, \mathcal{T}', \mathcal{A}' \rangle$ with $\mathcal{A}'$ being a reduced ABox such that KB has a pseudo-model (resp. model) iff KB' has a pseudo-model (resp. model).

3. KB can be converted in polynomial time in the size of $\mathcal{T}$ to a Horn-CPDL$_{reg}$ knowledge base $KB' = \langle \mathcal{R}, \mathcal{T}', \mathcal{A} \rangle$ with $\mathcal{T}'$ being a clausal Horn-CPDL$_{reg}$ TBox such that:

   - KB has a pseudo-model (resp. model) iff KB' has a pseudo-model (resp. model),
   - if $\mathcal{T}$ does not use the constructor $\forall R.C$ on the LHS of $\sqsubseteq$, then $\mathcal{T}'$ does neither.

Proof:
[Sketch] The first assertion is clear. For the second assertion, we start with $\mathcal{T}' := \mathcal{T}$ and $\mathcal{A}' := \mathcal{A}$ and then, for each $C(a) \in \mathcal{A}'$ where C is neither of the form A (a concept name) nor $\neg A$, we replace $C(a)$ in $\mathcal{A}'$ by $B(a)$, where B is a fresh concept name, and add to $\mathcal{T}'$ the axiom $B \sqsubseteq C$.

For the third assertion, we first transform roles occurring in $\mathcal{T}$ to the inverse normal form. We then replace every concept $\exists(R \circ S).C$ by $\exists R.\exists S.C$, and every concept $\exists(C?).D$ by $C \sqcap D$. After that we apply the technique that replaces complex concepts not of the form $\neg C$ by fresh concept names. For example, if $\forall R.C \sqsubseteq \exists S.D$ is an axiom of $\mathcal{T}$, where C and D are complex concepts, then we replace it by axioms $C \sqsubseteq A_C$, $\forall R.A_C \sqsubseteq \exists S.A_D$ and $A_D \sqsubseteq D$, where A_C and A_D are fresh concept names. In general, if a complex concept C occurs in a TBox axiom and is of the family C_l (resp. C_r) and is not the whole LHS (resp. RHS) of that axiom, then it is replaced by a fresh concept name A_C and the obtained TBox axiom is accompanied by the additional TBox axiom $C \sqsubseteq A_C$ (resp. $A_C \sqsubseteq C$). Furthermore, we replace a TBox axiom $C \sqsubseteq \neg D$ by $C \sqcap D \sqsubseteq \bot$, and $C \sqsubseteq \neg D \sqcup E$ by $C \sqcap D \sqsubseteq E$.

It is straightforward to prove that the described transformations result in a knowledge base satisfying the properties stated in the proposition. □

Corollary 4.7. Every Horn-CPDL$_{reg}$ knowledge base KB can be converted in polynomial time in the sizes of $\mathcal{T}$ and $\mathcal{A}$ to a clausal Horn-CPDL$_{reg}$ knowledge base $KB' = \langle \mathcal{R}, \mathcal{T}', \mathcal{A}' \rangle$ such that KB has a pseudo-model (resp. model) iff KB' has a pseudo-model (resp. model).

Proof:
We first apply the conversion mentioned in the second assertion of Proposition 4.6 to KB to obtain KB_2, and then apply the conversion mentioned in the third assertion of Proposition 4.6 to KB_2 to obtain KB'. □

4.3. Properties of the Constructive Semantics

We present below basic properties of the constructive semantics of Horn-CPDL$_{reg}$.

Theorem 4.8. Let KB be a clausal Horn-CPDL$_{reg}$ knowledge base and $C(a)$ be a Horn-CPDL$_{reg}$ query. Then:

1. If $KB \mid\!\!\sim C(a)$, then $KB \models C(a)$.

2. If $\{\top \sqsubseteq \exists R.\top \mid R \in \mathbf{R}\} \subseteq \mathcal{T}$, then:

(a) if KB has a pseudo-model, then it also has a model,

(b) $KB \mid\!\sim C(a)$ iff $KB \models C(a)$.

3. If KB is specified without using the constructor $\forall R_{l\forall}.C_l$ in the grammar rule (8) and has a pseudo-model, then it also has a model.

4. If KB and C are specified without using the constructor $\forall R_{l\forall}.C_l$ in the grammar rule (8), then $KB \mid\!\sim C(a)$ iff $KB \models C(a)$. □

The first two assertions follow from Remark 4.5. The third assertion is proved as Lemma A.5 in the Appendix. The last assertion follows immediately from Proposition 4.6 and the third assertion.

5. Checking Constructive Satisfiability in Horn-CPDL$_{reg}$

In this section we present an algorithm that, given a clausal Horn-CPDL$_{reg}$ knowledge base $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ together with the RIA-automaton-specification $\mathbf{A}$ of $\mathcal{R}$, checks whether the knowledge base has a pseudo-model.

5.1. Automaton-Modal Operators

We say that a role is in the inverse-and-test normal form (ITNF) if it is in the inverse normal form and uses the test operator $C?$ only for concepts C of the form A or $\neg A$. Such a role can be treated as a regular expression over the alphabet $\Sigma = \mathbf{R} \cup \{A?, (\neg A)? \mid A \in \mathbf{C}\}$ (where $\circ$ corresponds to ; and $\sqcup$ corresponds to $\cup$). The regular language characterized by such a role R is denoted by $\mathcal{L}(R)$. A word $R_1 R_2 \ldots R_k$ over Σ is also treated as the role $R_1 \circ R_2 \circ \cdots \circ R_k$.

For each role R in ITNF, let $\mathbb{A}_R$ be a finite automaton recognizing the regular language $\mathcal{L}(R)$. For each role R in ITNF such that $R \notin \mathbf{R}$, let $\mathbf{A}_R$ be a finite automaton recognizing the language $\mathcal{L}(R')$, where R' is obtained from R by simultaneously substituting each $S \in \mathbf{R}$ by a regular expression representing $\mathcal{L}(\mathbf{A}_S)$ (the set of all words accepted by $\mathbf{A}_S$).

The automaton $\mathbb{A}_R$ can be constructed from R in polynomial time, and $\mathbf{A}_R$ can be constructed in polynomial time in the sizes of R and the automata $(\mathbf{A}_S)_{S \in \mathbf{R}}$. Roughly speaking, $\mathbf{A}_R$ can be obtained from $\mathbb{A}_R$ by simultaneously substituting each transition $\langle q_1, S, q_2 \rangle$ by the automaton $\mathbf{A}_S$.

Given a role R in ITNF, by $\mathbf{A}_{\overline{R}}$ we denote $\mathbf{A}_S$ with S being $\overline{R}$ in ITNF.

We assume that for every used finite automaton, all of the states are *reachable* (from the initial state) and *productive* (i.e. from any of them we can reach a final state).

Given an interpretation (resp. pseudo-interpretation) $\mathcal{I}$ and a finite automaton A over alphabet Σ, we define $\mathsf{A}^{\mathcal{I}}$ (resp. $\mathsf{A}^{\mathcal{I}_\forall}$, $\mathsf{A}^{\mathcal{I}_\exists}$) to be $\{\langle x, y \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid$ there exist a word $R_1 \ldots R_k$ accepted by A and elements $x_0 = x, x_1, \ldots, x_k = y$ of $\Delta^{\mathcal{I}}$ such that $\langle x_{i-1}, x_i \rangle \in R_i^{\mathcal{I}}$ (resp. $\langle x_{i-1}, x_i \rangle \in R_i^{\mathcal{I}_\forall}$, $\langle x_{i-1}, x_i \rangle \in R_i^{\mathcal{I}_\exists}$) for all $1 \leq i \leq k\}$.

We will use auxiliary concept constructors $[\mathsf{A}]C$, $[\mathsf{A}]_\exists C$ and $\langle \mathsf{A} \rangle C$, where A is a finite automaton over alphabet Σ and C is a concept. Such constructors (called formulas with automaton-modal operators) were used earlier, among others, in [1, 20, 21, 7, 4]. The semantics of concepts $[\mathsf{A}]C$, $[\mathsf{A}]_\exists C$, $\langle \mathsf{A} \rangle C$ are specified below:

- given an interpretation $\mathcal{I}$,

$$\begin{aligned}([\mathsf{A}]C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y(\langle x,y\rangle \in \mathsf{A}^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}})\},\\ (\langle \mathsf{A}\rangle C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y(\langle x,y\rangle \in \mathsf{A}^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\};\end{aligned}$$

- given a pseudo-interpretation $\mathcal{I}$,

$$\begin{aligned}([\mathsf{A}]C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y(\langle x,y\rangle \in \mathsf{A}^{\mathcal{I}_\forall} \Rightarrow y \in C^{\mathcal{I}})\},\\ ([\mathsf{A}]_\exists C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y(\langle x,y\rangle \in \mathsf{A}^{\mathcal{I}_\exists} \Rightarrow y \in C^{\mathcal{I}})\},\\ (\langle \mathsf{A}\rangle C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y(\langle x,y\rangle \in \mathsf{A}^{\mathcal{I}_\exists} \wedge y \in C^{\mathcal{I}})\}.\end{aligned}$$

For a finite automaton A over Σ, let the components of A be denoted as in

$$\mathsf{A} = \langle \Sigma, Q_{\mathsf{A}}, q_{\mathsf{A}}, \delta_{\mathsf{A}}, F_{\mathsf{A}}\rangle.$$

If q is a state of a finite automaton A, then by A_q we denote the finite automaton obtained from A by replacing the initial state by q.

Lemma 5.1. Let $\mathcal{I}$ be a pseudo-model of a regular RBox $\mathcal{R}$, $\mathbf{A}$ the RIA-automaton-specification of $\mathcal{R}$, and C a concept. Then:

- $(\forall R.C)^{\mathcal{I}} = ([\mathbf{A}_R]C)^{\mathcal{I}}$ and $(\exists R.C)^{\mathcal{I}} = (\langle \mathbf{A}_R\rangle C)^{\mathcal{I}}$,
- $C^{\mathcal{I}} \subseteq ([\mathbf{A}_{\overline{R}}]_\exists \langle \mathbf{A}_R\rangle C)^{\mathcal{I}}$ and $C^{\mathcal{I}} \subseteq ([\mathbf{A}_{\overline{R}}]_\exists \exists R.C)^{\mathcal{I}}$. □

The proof of this lemma is straightforward.

5.2. Our Algorithm

Let X be a set of concepts. The *saturation* of X (w.r.t. $\mathbf{A}$ and $\mathcal{T}$), denoted by $\mathsf{Satr}(X)$, is defined to be the least extension of X such that:

1. for every $R \in \mathbf{R}$, $[\mathbf{A}_{\overline{R}}]_\exists \exists R.\top \in \mathsf{Satr}(X)$,
2. if $\forall R.C \in \mathsf{Satr}(X)$, then $[\mathbf{A}_R]C \in \mathsf{Satr}(X)$,
3. if $[\mathsf{A}]C \in \mathsf{Satr}(X)$, $\langle q_{\mathsf{A}}, B?, q\rangle \in \delta_{\mathsf{A}}$ and $B \in \mathsf{Satr}(X)$, then $[\mathsf{A}_q]C \in \mathsf{Satr}(X)$,
4. if $[\mathsf{A}]_\exists C \in \mathsf{Satr}(X)$, $\langle q_{\mathsf{A}}, B?, q\rangle \in \delta_{\mathsf{A}}$ and $B \in \mathsf{Satr}(X)$, then $[\mathsf{A}_q]_\exists C \in \mathsf{Satr}(X)$,
5. if ($[\mathsf{A}]C \in \mathsf{Satr}(X)$ or $[\mathsf{A}]_\exists C \in \mathsf{Satr}(X)$) and $q_{\mathsf{A}} \in F_{\mathsf{A}}$, then $C \in \mathsf{Satr}(X)$,
6. if $B \in \mathsf{Satr}(X)$ and $\exists R.B$ occurs on the LHS of $\sqsubseteq$ in some clause of $\mathcal{T}$, then $[\mathbf{A}_{\overline{R}}]_\exists \langle \mathbf{A}_R\rangle B \in \mathsf{Satr}(X)$.[3]

[3] As stated in Section 2, the letter B denotes a concept name (i.e., an atomic concept).

For $R \in \mathbf{R}$, there are two kinds of *transfer* of X through R:

$$\begin{array}{rcl} \mathsf{Trans}(X,R) & = & \{[\mathsf{A}_q]C \mid [\mathsf{A}]C \in X \text{ and } \langle q_{\mathsf{A}}, R, q\rangle \in \delta_{\mathsf{A}}\} \\ \mathsf{Trans}_{\exists}(X,R) & = & \mathsf{Trans}(X,R) \cup \{[\mathsf{A}_q]_{\exists}C \mid [\mathsf{A}]_{\exists}C \in X \text{ and } \langle q_{\mathsf{A}}, R, q\rangle \in \delta_{\mathsf{A}}\}. \end{array}$$

Our algorithm for checking whether $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ has a pseudo-model uses the data structure $G = \langle \Delta_0, \Delta, \mathit{Label}, \mathit{Next}, \mathit{LeastSucc}, \mathit{Status}\rangle$, which is called a ***Horn-CPDL$_{reg}$ graph***, where:

- Δ_0 : the set of all individual names occurring in $\mathcal{A}$,
- Δ : a set of individuals such that $\Delta_0 \subseteq \Delta$,
- Label : a function mapping each $x \in \Delta$ to a set of concepts,
- $\mathit{Next} : \Delta \times \{\exists R.\top, \exists R.A \mid R \in \mathbf{R}, A \in \mathbf{C}\} \to \Delta$ is a partial mapping,
- $\mathit{LeastSucc} : \Delta \times \mathbf{R} \to \Delta$ is a partial mapping,
- $\mathit{Status} \in \{\mathit{unknown}, \mathit{unsat}, \mathit{sat}\}$.

Define $\mathit{Edges} = \{\langle x, R, y\rangle \mid R(x,y) \in \mathcal{A}$ or $\mathit{Next}(x, \exists R.C) = y$ for some C or $\mathit{LeastSucc}(x,R) = y\}$. A tuple $\langle x, R, y\rangle \in \mathit{Edges}$ represents an edge $\langle x, y\rangle$ labeled by R of the graph. If $R(x,y) \in \mathcal{A}$ or $\mathit{Next}(x, \exists R.C) = y$, then we call $\langle x, R, y\rangle$ a *firm edge*, else if $\mathit{LeastSucc}(x,R) = y$, then we call $\langle x, R, y\rangle$ a *pseudo edge*. The notions of *predecessor* and *successor* are defined as usual. We say that $x \in \Delta$ is *reachable* from Δ_0 if there exist $x_0, \ldots, x_k \in \Delta$ and elements $R_1, \ldots, R_k$ of $\mathbf{R}$ such that $k \geq 0$, $x_0 \in \Delta_0$, $x_k = x$ and $\langle x_{i-1}, R_i, x_i\rangle \in \mathit{Edges}$ for all $1 \leq i \leq k$.

For $x \in \Delta$, $\mathit{Label}(x)$ is called the *label* of x. A fact $\mathit{Next}(x, \exists R.C) = y$ means that $\exists R.C \in \mathit{Label}(x)$, $C \in \mathit{Label}(y)$, and $\exists R.C$ is "realized" at x by going to y. When defined, $\mathit{Next}(x, \exists R.\top)$ denotes the "logically smallest firm R-successor of x", and $\mathit{LeastSucc}(x,R)$ denotes the "logically smallest R-successor of x". A fact $\mathit{Status} = \mathit{unsat}$ means the knowledge base does not have any pseudo-model. A fact $\mathit{Status} = \mathit{sat}$ means the knowledge base has a pseudo-model.

Definition 5.2. Let $G, x \not\models_c [\mathsf{A}]B$ stand for "it is not certain that G satisfies $[\mathsf{A}]B$ at x", where $x \in \Delta$, A is a finite automaton over Σ and $B \in \mathbf{C}$. We define $\not\models_c$ to be the smallest relation such that $G, x \not\models_c [\mathsf{A}]B$ holds if one of the following holds (for some B' or R when it is related):

- $q_{\mathsf{A}} \in F_{\mathsf{A}}$ and $B \notin \mathit{Label}(x)$;
- $\langle q_{\mathsf{A}}, (\neg B')?, q\rangle \in \delta_{\mathsf{A}}$, $B' \notin \mathit{Label}(x)$ and $G, x \not\models_c [\mathsf{A}_q]B$;
- $\langle q_{\mathsf{A}}, R, q\rangle \in \delta_{\mathsf{A}}$, $\exists R.\top \notin \mathit{Label}(x)$ and $\mathit{LeastSucc}(x,R)$ is not defined;
- $\langle q_{\mathsf{A}}, R, q\rangle \in \delta_{\mathsf{A}}$, $\exists R.\top \notin \mathit{Label}(x)$, $\mathit{LeastSucc}(x,R) = y$ and $G, y \not\models_c [\mathsf{A}_q]B$;
- $\langle q_{\mathsf{A}}, R, q\rangle \in \delta_{\mathsf{A}}$, $\exists R.\top \in \mathit{Label}(x)$ and $\mathit{Next}(x, \exists R.\top)$ is not defined;
- $\langle q_{\mathsf{A}}, R, q\rangle \in \delta_{\mathsf{A}}$, $\mathit{Next}(x, \exists R.\top) = y$ and $G, y \not\models_c [\mathsf{A}_q]B$.

We define that $G, x \not\models_c \forall R.A$ if $G, x \not\models_c [\mathbf{A}_R]A$. □

Function Find(X)

if *there exists* $z \in \Delta \setminus \Delta_0$ *with* $Label(z) = X$ **then return** z;
add a new element z to Δ with $Label(z) := X$;
return z;

Procedure ExtendLabel(z, X)

if $X \subseteq Label(z)$ **then return**;
if $z \in \Delta_0$ **then** $Label(z) := \mathsf{Satr}(Label(z) \cup X)$;
else
    $z_* := \mathtt{Find}(\mathsf{Satr}(Label(z) \cup X))$;
    foreach y, R, C *such that* $Next(y, \exists R.C) = z$ **do** $Next(y, \exists R.C) := z_*$;
    foreach y *and* R *such that* $LeastSucc(y, R) = z$ **do** $LeastSucc(y, R) := z_*$;

Function CheckPremise(x, C)

if $C = \top$ **then return** $true$;
else let $C = C_1 \sqcap \ldots \sqcap C_k$;
foreach $1 \leq i \leq k$ **do**
    if $C_i = A$ *and* $A \notin Label(x)$ **then return** $false$;
    else if $C_i = \forall\exists R.A$ *and* ($\exists R.\top \notin Label(x)$ *or* $Next(x, \exists R.\top)$ *is not defined or* $A \notin Label(Next(x, \exists R.\top))$) **then return** $false$;
    else if $C_i = \exists R.A$ *and* $\langle \mathbf{A}_R \rangle A \notin Label(x)$ **then return** $false$;
    else if $C_i = \forall R.A$ *and* $G, x \not\models_c \forall R.A$ **then return** $false$;
return $true$;

Algorithm 1: checking constructive satisfiability in Horn-CPDL$_{reg}$

Input: a clausal Horn-CPDL$_{reg}$ knowledge base $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ and the RIA-automaton-specification $\mathbf{A}$ of $\mathcal{R}$.
Output: $true$ if KB has a pseudo-model, or $false$ otherwise.
Data structure: a Horn-CPDL$_{reg}$ graph $G = \langle \Delta_0, \Delta, Label, Next, LeastSucc, Status \rangle$.

let Δ_0 be the set of all individuals occurring in $\mathcal{A}$;
if $\Delta_0 = \emptyset$ **then** $\Delta_0 := \{\tau\}$;
$\Delta := \Delta_0$, $Status := unknown$;
set $Next$ and $LeastSucc$ to the empty mappings;
foreach $a \in \Delta_0$ **do**
    $Label(a) := \mathsf{Satr}(\{A \mid A(a) \in \mathcal{A}\} \cup \mathcal{T})$;
while *some rule in Table 1 can make changes* **do**
    choose such a rule and execute it; `// any strategy can be used`
    if $Status = unsat$ **then return** $false$;
return $true$;

$(\forall_1)$	**if** $r(a,b) \in \mathcal{A}$ **then** $\texttt{ExtendLabel}(b, \mathsf{Trans}_\exists(\mathit{Label}(a), r))$, $\texttt{ExtendLabel}(a, \mathsf{Trans}_\exists(\mathit{Label}(b), \overline{r}))$;
$(\forall_2)$	**if** x is reachable from Δ_0 and $\mathit{Next}(x, \exists R.C) = y$ **then** $\mathit{Next}(x, \exists R.C) := \texttt{Find}(\mathsf{Satr}(\mathit{Label}(y) \cup \mathsf{Trans}_\exists(\mathit{Label}(x), R)))$;
$(\forall_3)$	**if** x is reachable from Δ_0 and $\mathit{Next}(x, \exists R.C) = y$ **then** $\texttt{ExtendLabel}(x, \mathsf{Trans}_\exists(\mathit{Label}(y), \overline{R}))$;
$(\forall_4)$	**if** x is reachable from Δ_0 and $\mathit{LeastSucc}(x, R) = y$ **then** $\mathit{LeastSucc}(x, R) := \texttt{Find}(\mathsf{Satr}(\mathit{Label}(y) \cup \mathsf{Trans}(\mathit{Label}(x), R)))$;
$(\forall_5)$	**if** x is reachable from Δ_0 and $\mathit{LeastSucc}(x, R) = y$ **then** $\texttt{ExtendLabel}(x, \mathsf{Trans}(\mathit{Label}(y), \overline{R}))$;
$(\exists)$	**if** x is reachable from Δ_0, $\exists R.C \in \mathit{Label}(x)$, $R \in \mathbf{R}$ and $\mathit{Next}(x, \exists R.C)$ is not defined **then** $\mathit{Next}(x, \exists R.C) := \texttt{Find}(\mathsf{Satr}(\{C\} \cup \mathsf{Trans}_\exists(\mathit{Label}(x), R) \cup \mathcal{T}))$;
(LS)	**if** x is reachable from Δ_0, $R \in \mathbf{R}$ and $\mathit{LeastSucc}(x, R)$ is not defined **then** $\mathit{LeastSucc}(x, R) := \texttt{Find}(\mathsf{Satr}(\mathsf{Trans}(\mathit{Label}(x), R) \cup \mathcal{T}))$;
$(\sqsubseteq)$	**if** x is reachable from Δ_0, $(C \sqsubseteq D) \in \mathit{Label}(x)$ and $\texttt{CheckPremise}(x, C)$ **then** $\texttt{ExtendLabel}(x, \{D\})$;
$(\bot)$	**if** $\bot \in \mathit{Label}(x)$ or there exists $\{A, \neg A\} \subseteq \mathit{Label}(x)$ **then** $\mathit{Status} := \mathit{unsat}$;

Table 1. Expansion rules for Horn-CPDL$_{reg}$ graphs.

Algorithm 1 (on page 127) attempts to construct a pseudo-model of KB by initializing a Horn-CPDL$_{reg}$ graph and then expanding it by the rules in Table 1 (on page 128). The intended pseudo-model extends $\mathcal{A}$ with disjoint trees rooted at the named individuals occurring in $\mathcal{A}$. The trees may be infinite. However, we represent such a semi-forest as a finite graph with global caching: if two nodes that are not named individuals occur in a tree or in different trees and have the same label, then they should be merged. In other words, for every finite set X of concepts, the graph contains at most one node $z \in \Delta \setminus \Delta_0$ such that $\mathit{Label}(z) = X$. The function $\texttt{Find}(X)$ returns such a node z if it exists, or creates such a node z otherwise.

For each $x \in \Delta$, $\mathit{Label}(x)$ is a set of requirements to be "realized" at x. To realize such requirements at nodes, sometimes we have to extend their labels. Suppose we want to extend the label of $z \in \Delta$ with a set X of concepts. Consider the following cases:

- Case $z \in \Delta_0$ (i.e., z is a named individual occurring in $\mathcal{A}$): as z is "fixed" by the ABox $\mathcal{A}$, we have no choice but to extend $\mathit{Label}(z)$ directly with $\mathsf{Satr}(X)$.
- Case $z \notin \Delta_0$ and the requirements X are directly caused by z itself or its successors: if we directly extend the label of z (with $\mathsf{Satr}(X)$), then z will possibly have the same label as another node not belonging to Δ_0 and global caching is not fulfilled. Hence, we "simulate" changing the label of

z by using $z_* := \texttt{Find}(\mathsf{Satr}(\mathit{Label}(z) \cup X))$ for playing the role of z. In particular, for each y, R and C such that $\mathit{Next}(y, \exists R.C) = z$, we set $\mathit{Next}(y, \exists R.C) := z_*$, and for each y and R such that $\mathit{LeastSucc}(y, R) = z$, we set $\mathit{LeastSucc}(y, R) := z_*$.

Extending the label of z for the above two cases is done by Procedure $\texttt{ExtendLabel}(z, X)$. The expansion rules $(\forall_1)$, $(\forall_3)$, $(\forall_5)$ (in Table 1) deal with these two cases. The third case is considered below.

Suppose that $\mathit{Next}(x, \exists R.C) = y$. Then, to realize the requirements at x, the label of y should be extended with $X = \mathsf{Trans}_\exists(\mathit{Label}(x), R)$. How can we realize such an extension? Recall that we intend to construct a forest-like model for KB, but use global caching to guarantee termination. There may exist another $\mathit{Next}(x', \exists R'.C') = y$ with $x' \neq x$. That is, we may use y as a successor for two different nodes x and x', but the intention is to put x and x' into disjoint trees. If we directly modify the label of y to realize the requirements of x, such a modification may affect x'. The solution is to delete the edge $\langle x, R, y\rangle$ and reconnect x to $y_* := \texttt{Find}(\mathit{Label}(y) \cup X)$ by setting $\mathit{Next}(x, \exists R.C) := y_*$. The extension is formally realized by the expansion rule $(\forall_2)$ (in Table 1). A similar case concerning $\mathit{LeastSucc}(x, R) = y$ is dealt with by the expansion rule $(\forall_4)$.

Consider the other expansion rules (in Table 1):

- $(\exists)$: If $\exists R.C \in \mathit{Label}(x)$ and $\mathit{Next}(x, \exists R.C)$ is not defined yet, then, to realize the requirement $\exists R.C$ at x, we connect x by a firm edge via R to a node with label $X = \mathsf{Satr}(\{C\} \cup \mathsf{Trans}_\exists(\mathit{Label}(x), R) \cup \mathcal{T})$ by setting $\mathit{Next}(x, \exists R.C) := \texttt{Find}(X)$.

- (LS): If $R \in \mathbf{R}$ and $\mathit{LeastSucc}(x, R)$ is not defined yet, then we connect x by a pseudo edge via R to a node with label $X = \mathsf{Satr}(\mathsf{Trans}(\mathit{Label}(x), R) \cup \mathcal{T})$ by setting $\mathit{LeastSucc}(x, R) := \texttt{Find}(X)$.

- $(\sqsubseteq)$: If $(C \sqsubseteq D) \in \mathit{Label}(x)$ and C "holds" at x, then we extend the label of x with $\{D\}$ by using the procedure ExtendLabel discussed earlier. Suppose $C = C_1 \sqcap \ldots \sqcap C_k$. How to check whether C "holds" at x? It "holds" at x if C_i "holds" at x for each $1 \leq i \leq k$. There are the following cases:

  - Case $C_i = A$: C_i "holds" at x if $A \in \mathit{Label}(x)$.
  - Case $C_i = \forall\exists R.A$ with $R \in \mathbf{R}$: C_i "holds" at x if both $\forall R.A$ and $\exists R.\top$ "hold" at x. If $\exists R.\top$ "holds" at x because the automaton $\mathbf{A}_R$ accepts a word $S_1 \ldots S_h$ and there is a path of nodes $x_0, \ldots, x_h$ such that $x_0 = x$ and for each $1 \leq j \leq h$, either $\langle x_{j-1}, S_j, x_j\rangle$ or $\langle x_j, \overline{S}_j, x_{j-1}\rangle$ is a firm edge (belonging to Edges), then:
    - the automaton $\mathsf{A} = \mathbf{A}_{\overline{R}}$ accepts $\overline{S}_h \ldots \overline{S}_1$ by a run $q_h, \ldots, q_0$;
    - since $\mathit{Label}(x_h)$ is saturated, $[\mathbf{A}_{\overline{R}}]_\exists \exists R.\top \in \mathit{Label}(x_h)$, which means $[\mathsf{A}_{q_h}]_\exists \exists R.\top \in \mathit{Label}(x_h)$;
    - by the rules $(\forall_1)$-$(\forall_3)$, for each j from $h-1$ to 0, we can expect that $[\mathsf{A}_{q_j}]_\exists \exists R.\top \in \mathit{Label}(x_j)$;
    - consequently, we can expect that $[\mathsf{A}_{q_0}]_\exists \exists R.\top \in \mathit{Label}(x_0)$;
    - since $x = x_0$ and $q_0 \in F_\mathsf{A}$, due to the saturation, we can expect that $\exists R.\top \in \mathit{Label}(x)$ and $\mathit{Next}(x, \exists R.\top)$ is defined.

    Thus, to check whether C_i "holds" at x we just check whether $\exists R.\top \in \mathit{Label}(x)$, $\mathit{Next}(x, \exists R.\top)$ is defined and $A \in \mathit{Label}(\mathit{Next}(x, \exists R.\top))$. The intuition is that $y =$

$Next(x, \exists R.\top)$ is the "logically smallest firm R-successor of x", and if $A \in Label(y)$ then A will occur in all firm R-successors of x.

- Case $C_i = \exists R.A$: If $\exists R.A$ "holds" at x because the automaton $\mathbf{A}_R$ accepts a word $S_1 \dots S_h$ and there is a path of nodes $x_0, \dots, x_h$ such that $x_0 = x$, $A \in Label(x_h)$ and, for each $1 \leq j \leq h$:
  - either $\langle x_{j-1}, S_j, x_j \rangle$ or $\langle x_j, \overline{S}_j, x_{j-1} \rangle$ is a firm edge,
  - or $x_{j-1} = x_j$, S_j has the form $B?$ and $B \in Label(x_j)$,

  then:
  - the automaton $\mathsf{A} = \mathbf{A}_{\overline{R}}$ accepts $\overline{S}_h \dots \overline{S}_1$ by a run $q_h, \dots, q_0$;
  - since $Label(x_h)$ is saturated, $[\mathbf{A}_{\overline{R}}]_\exists \langle \mathbf{A}_R \rangle A \in Label(x_h)$, which means $[\mathsf{A}_{q_h}]_\exists \langle \mathbf{A}_R \rangle A \in Label(x_h)$;
  - by the rules $(\forall_1)$-$(\forall_3)$ and the saturation, for each j from $h-1$ to 0, we can expect that $[\mathsf{A}_{q_j}]_\exists \langle \mathbf{A}_R \rangle A \in Label(x_j)$;
  - consequently, we can expect that $[\mathsf{A}_{q_0}]_\exists \langle \mathbf{A}_R \rangle A \in Label(x_0)$;
  - since $x = x_0$ and $q_0 \in F_{\mathsf{A}}$, due to the saturation, we can expect that $\langle \mathbf{A}_R \rangle A \in Label(x)$.

  Thus, to check whether $C_i = \exists R.A$ "holds" at x, we just check whether $\langle \mathbf{A}_R \rangle A \in Label(x)$. (Semantically, $\langle \mathbf{A}_R \rangle A$ is equivalent to $\exists R.A$.)
- Case $C_i = \forall R.A$: C_i "holds" at x if it does not hold that $G, x \not\models_c \forall R.A$.

Formally, checking whether the premise C of a Horn-CPDL$_{reg}$ clause $C \sqsubseteq D$ "holds" at x is done by Function `CheckPremise`(x, C).

- $(\bot)$: If $Label(x)$ contains $\bot$ or a pair $A, \neg A$, then $Status$ is set to $unsat$, which causes Algorithm 1 to be terminated with the result $false$, which means that the knowledge base KB does not have any pseudo-model.

We do global caching to represent a possibly infinite semi-forest by a finite graph possibly with cycles. As a side effect, direct checking "realization" of existential automaton-modal operators is not safe. Furthermore, we cannot allow universal modal operators to "run" along such cycles. "Running" universal modal operators backward along an edge is safe, but "running" universal modal operators forward along an edge is done using a special technique, which may replace the edge by another one as in the rules $(\forall_2)$ and $(\forall_4)$.

Expansions by modifying the label of a node and/or setting the mapping $Next$ or $LeastSucc$ are done only for nodes that are reachable from Δ_0. Note that, when a node z is simulated by z_* as in Procedure `ExtendLabel`, the node z becomes unreachable from Δ_0. We do not delete such nodes z because they may be reused later.

When the graph cannot be expanded any more, the algorithm terminates in the normal mode with result $true$, which means KB has a pseudo-model.

Theorem 5.3. Algorithm 1 runs in polynomial time in the size of the ABox $\mathcal{A}$ and correctly checks whether the clausal Horn-CPDL$_{reg}$ knowledge base KB has a pseudo-model. □

This theorem follows immediately from Lemmas A.1, A.2 and Corollary A.4, which are given and proved in the Appendix. The following corollary follows from this theorem and Proposition 4.6.

Corollary 5.4. The instance checking problem in Horn-CPDL$_{reg}$ with respect to the constructive semantics has PTIME data complexity. □

5.3. An Illustrative Example

Example 5.5. Let $\mathbf{R}_+ = \{r, s\}$, $\mathbf{C} = \{A, B, C, D, E\}$, $\mathbf{I} = \{a, b\}$, $\mathcal{R} = \{\overline{r} \circ r \sqsubseteq r, \overline{r} \circ r \sqsubseteq \overline{r}\}$, and let $\mathcal{T}$ be the TBox consisting of the following axioms:

$$A \sqsubseteq \exists r.C \tag{15}$$
$$C \sqsubseteq \forall \overline{r}.D \tag{16}$$
$$D \sqsubseteq C \tag{17}$$
$$A \sqcap \forall \exists r.C \sqsubseteq E \tag{18}$$
$$A \sqcap \exists r.B \sqsubseteq E \tag{19}$$
$$E \sqsubseteq \forall (s \sqcup \overline{s})^*.F \tag{20}$$
$$\forall s^*.F \sqsubseteq \bot. \tag{21}$$

Like Example 2.1, $\mathcal{R}$ is a regular RBox with the following RIA-automaton-specification:

$$\begin{aligned}
\mathbf{A}_r &= \langle \mathbf{R}, \{0,1,2\}, 0, \{\langle 0, r, 1\rangle, \langle 0, \overline{r}, 2\rangle, \langle 2, r, 2\rangle, \langle 2, \overline{r}, 2\rangle, \langle 2, r, 1\rangle\}, \{1\}\rangle \\
\mathbf{A}_{\overline{r}} &= \langle \mathbf{R}, \{0,1,2\}, 0, \{\langle 0, \overline{r}, 1\rangle, \langle 0, \overline{r}, 2\rangle, \langle 2, r, 2\rangle, \langle 2, \overline{r}, 2\rangle, \langle 2, r, 1\rangle\}, \{1\}\rangle \\
\mathbf{A}_s &= \langle \mathbf{R}, \{0,1\}, 0, \{\langle 0, s, 1\rangle\}, \{1\}\rangle \\
\mathbf{A}_{\overline{s}} &= \langle \mathbf{R}, \{0,1\}, 0, \{\langle 0, \overline{s}, 1\rangle\}, \{1\}\rangle.
\end{aligned}$$

We also have that:

$$\begin{aligned}
\mathbf{A}_{s^*} &= \langle \Sigma, \{0\}, 0, \{\langle 0, s, 0\rangle\}, \{0\}\rangle \\
\mathbf{A}_{(s \sqcup \overline{s})^*} &= \langle \Sigma, \{0\}, 0, \{\langle 0, s, 0\rangle, \langle 0, \overline{s}, 0\rangle\}, \{0\}\rangle.
\end{aligned}$$

Consider the clausal Horn-CPDL$_{reg}$ knowledge base $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ with $\mathcal{A} = \{A(a),\ B(a),\ A(b),\ r(a,b)\}$. Figure 4 illustrates the Horn-CPDL$_{reg}$ graph constructed by Algorithm 1 for KB. The nodes of the graph are a, b, u, u', v, v', w, where $\Delta_0 = \{a, b\}$. In each node, we display the concepts of the label of the node. The main steps of the run of the algorithm are numbered from 0 to 15. In the table representing a node $x \in \{a, b\}$, the number in the left cell in a row denotes the step at which the concepts in the right cell were added to the label of the node. For a node not belonging to $\Delta_0 = \{a, b\}$, the number before the name of the node denotes the step at which the node was created. A label $n : \exists r.\varphi$ displayed for an edge from a node x to a node y means that $Next(x, \exists r.\varphi) = y$ and the edge was created at the step n. A label $n : s$ displayed for an edge from a node x to a node y means that $LeastSucc(x, s) = y$ and the edge was created at the step n. A label $n : deleted$ beside a dashed edge means that the edge was deleted at the step n.

The steps of running Algorithm 1 for KB are as follows:

0: initialization,

1: applying the expansion rule ($\sqsubseteq$) to the node $x = a$ using the clause (15),

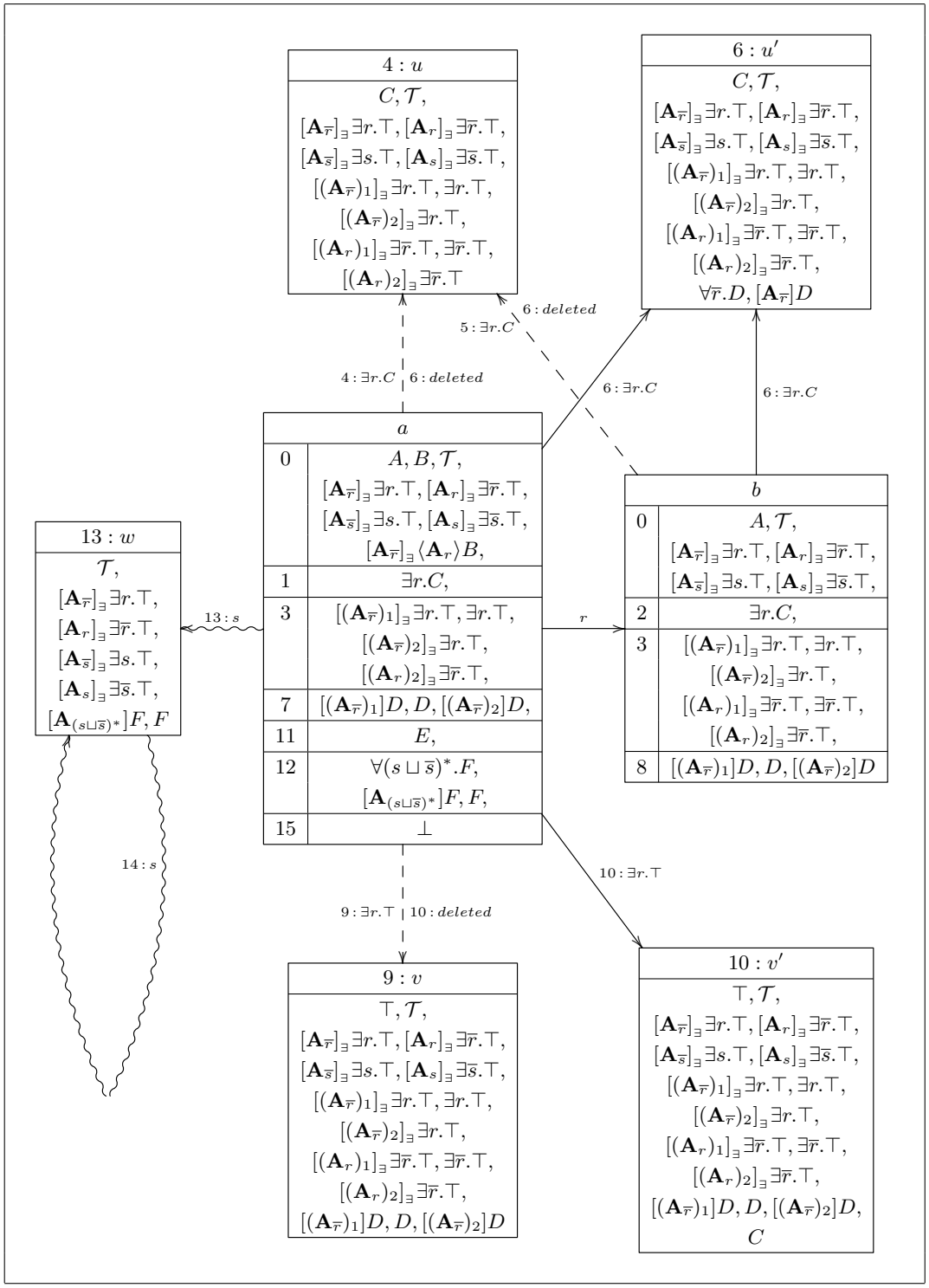


Figure 4. An illustration for Example 5.5.

2: applying ($\sqsubseteq$) to the node $x = b$ using the clause (15),
3: applying ($\forall_1$) to the nodes a and b twice,
4: applying ($\exists$) to the node $x = a$ and the concept $\exists r.C$,
5: applying ($\exists$) to the node $x = b$ and the concept $\exists r.C$,
6: applying ($\sqsubseteq$) to the node $x = u$ using the clause (16),
7: applying ($\forall_2$) to the nodes $x = a$ and $y = u'$,
8: applying ($\forall_2$) to the nodes $x = b$ and $y = u'$,
9: applying ($\exists$) to the node $x = a$ and the concept $\exists r.\top$,
10: applying ($\sqsubseteq$) to the node $x = v$ using the clause (17),
11: applying ($\sqsubseteq$) to the node $x = a$ using the clause (18),
12: applying ($\sqsubseteq$) to the node $x = a$ using the clause (20),
13: applying (LS) to the node $x = a$ and the role $R = s$,
14: applying (LS) to the node $x = w$ and the role $R = s$,
15: applying ($\sqsubseteq$) to the node $x = a$ using the clause (21).

Since $\bot$ was added to $Label(a)$, Algorithm 1 returns $false$, and by Corollary A.4, the knowledge base KB does not have any pseudo-model. □

6. Concluding Remarks

We have developed the rule language Horn-CPDL$_{reg}$ and proved that the instance checking problem in this language with respect to the constructive semantics has PTIME data complexity by providing an algorithm for checking whether a given knowledge base in Horn-CPDL$_{reg}$ has a pseudo-model.

Horn-CPDL$_{reg}$ is more general than the Horn fragments introduced and studied in our (joint) works [24, 20, 21, 22]. As it is tractable and more general than Horn-TEAMLOG [22], it is a useful rule language for formalizing agents' cooperation.

In contrast to all the well-known Horn fragments $\mathcal{EL}$ [9, 10], DL-Lite [17], DLP [8], Horn-$\mathcal{SHIQ}$ [11], Horn-$\mathcal{SROIQ}$ [15] of DLs, Horn-CPDL$_{reg}$ allows the concept constructors $\forall\exists R.C$ (for $R \in \mathbf{R}$) and $\forall R.C$ (for any role R) to appear on the LHS of TBox axioms.

In comparison with Horn-DL [25], apart from the concept constructor $\forall\exists R.C$ (for $R \in \mathbf{R}$), Horn-CPDL$_{reg}$ also allows the concept constructor $\forall R.C$ (for any role R) to appear on the LHS of TBox axioms. However, Horn-CPDL$_{reg}$ is not more general than Horn-DL because the latter additionally allows nominals, qualified number restrictions, the $\exists r.\mathsf{Self}$ constructor, the universal role as well as assertions of the form $\mathsf{disjoint}(s, s')$, $\mathsf{irreflexive}(s)$, $\neg s(a, b)$, $a \neq b$. As future work, we will extend Horn-CPDL$_{reg}$ with these features to obtain a rule language Horn-DL2 that is more general than Horn-DL, and hence also more general than Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SROIQ}$.

Our approach and method for Horn-CPDL$_{reg}$ make important steps in developing richer and richer tractable rule languages in modal and description logics.

Acknowledgements

This work was supported by the Polish National Science Centre (NCN) under Grant No. 2011/01/B/ST6/02769. We would like to thank the anonymous reviewers for helpful comments.

References

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

[2] Demri S. The Complexity of Regularity in Grammar Logics and Related Modal Logics. Journal of Logic and Computation. 2001;11(6):933–960.

[3] Demri S, de Nivelle H. Deciding Regular Grammar Logics with Converse Through First-Order Logic. Journal of Logic, Language and Information. 2005;14(3):289–329.

[4] Dunin-Kęplicz B, Nguyen LA, Szałas A. Converse-PDL with Regular Inclusion Axioms: A Framework for MAS Logics. J Applied Non-Classical Logics. 2011;21(1):61–91.

[5] Dunin-Kęplicz B, Verbrugge R. Collective Intentions. Fundam Inform. 2002;51(3):271–295.

[6] Dziubiński M, Verbrugge R, Dunin-Kęplicz B. Complexity Issues in Multiagent Logics. Fundam Inform. 2007;75(1-4):239–262.

[7] Nguyen LA. Horn Knowledge Bases in Regular Description Logics with PTime Data Complexity. Fundamenta Informaticae. 2010;104(4):349–384.

[8] Grosof BN, Horrocks I, Volz R, Decker S. Description logic programs: combining logic programs with description logic. In: Proceedings of WWW'2003; 2003. p. 48–57.

[9] Baader F, Brandt S, Lutz C. Pushing the $\mathcal{EL}$ Envelope. In: Proceedings of IJCAI'2005. Morgan-Kaufmann Publishers; 2005. p. 364–369.

[10] Baader F, Brandt S, Lutz C. Pushing the $\mathcal{EL}$ Envelope Further. In: Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions; 2008. .

[11] Hustadt U, Motik B, Sattler U. Reasoning in Description Logics by a Reduction to Disjunctive Datalog. J Autom Reasoning. 2007;39(3):351–384.

[12] Krisnadhi A, Lutz C. Data Complexity in the $\mathcal{EL}$ Family of Description Logics. In: Proceedings of LPAR'2007. vol. 4790 of LNCS. Springer; 2007. p. 333–347.

[13] Krötzsch M, Rudolph S, Hitzler P. Conjunctive Queries for a Tractable Fragment of OWL 1.1. In: Proceedings of ISWC'2007 + ASWC'2007, LNCS 4825. Springer; 2007. p. 310–323.

[14] Rosati R. On Conjunctive Query Answering in $\mathcal{EL}$. In: Proceedings of DL'2007;. p. 451–458.

[15] Ortiz M, Rudolph S, Simkus M. Query Answering in the Horn Fragments of the Description Logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In: Proceedings of IJCAI 2011; 2011. p. 1039–1044.

[16] Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R. Data complexity of query answering in description logics. Artif Intell. 2013;195:335–360.

[17] Calvanese D, Giacomo GD, Lembo D, Lenzerini M, Rosati R. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. J Autom Reasoning. 2007;39(3):385–429.

[18] Krötzsch M, Rudolph S, Hitzler P. Complexity Boundaries for Horn Description Logics. In: Proceedings of AAAI'2007. AAAI Press; 2007. p. 452–457.

[19] Brandt S. Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms, and - What Else? In: Proceedings of ECAI'2004. IOS Press; 2004. p. 298–302.

[20] Nguyen LA. Constructing Finite Least Kripke Models for Positive Logic Programs in Serial Regular Grammar Logics. Logic Journal of the IGPL. 2008;16(2):175–193.

[21] Dunin-Kęplicz B, Nguyen LA, Szałas A. Tractable Approximate Knowledge Fusion Using the Horn Fragment of Serial Propositional Dynamic Logic. Int J Approx Reasoning. 2010;51(3):346–362.

[22] Dunin-Kęplicz B, Nguyen LA, Szałas A. Horn-TeamLog: A Horn Fragment of TeamLog with PTime Data Complexity. In: Proceedings of ICCCI'2013. vol. 8083 of LNCS. Springer; 2013. p. 143–153.

[23] Nguyen LA. On the Deterministic Horn Fragment of Test-free PDL. In: Hodkinson I, Venema Y, editors. Advances in Modal Logic - Volume 6. King's College Publications; 2006. p. 373–392.

[24] Nguyen LA. A Bottom-up Method for the Deterministic Horn Fragment of the Description Logic $\mathcal{ALC}$. In: Proceedings of JELIA'2006. vol. 4160 of LNAI. Springer-Verlag; 2006. p. 346–358.

[25] Nguyen LA, Nguyen TBL, Szałas A. Towards richer rule languages with polynomial data complexity for the Semantic Web. Data Knowl Eng. 2015;96:57–77.

[26] Horrocks I, Kutz O, Sattler U. The Even More Irresistible $\mathcal{SROIQ}$. In: Proceedings of KR'2006. AAAI Press; 2006. p. 57–67.

A. Proofs

Define $closure_{\mathbf{A}}(\mathcal{T})$ for a TBox $\mathcal{T}$ to be the smallest set of concepts such that:

- concepts and subconcepts occurring in $\mathcal{T}$ belong to $closure_{\mathbf{A}}(\mathcal{T})$,
- subconcepts occurring in $closure_{\mathbf{A}}(\mathcal{T})$ also belong to $closure_{\mathbf{A}}(\mathcal{T})$,
- $\{[\mathbf{A}_{\overline{R}}]_{\exists}\exists R.\top \mid R \in \mathbf{R}\} \subseteq closure_{\mathbf{A}}(\mathcal{T})$,
- if $\exists R.B$ occurs on the LHS of $\sqsubseteq$ in a clause of $\mathcal{T}$, then $[\mathbf{A}_{\overline{R}}]_{\exists}\langle\mathbf{A}_R\rangle B \in \mathsf{Satr}(X)$,
- if $\forall R.C \in closure_{\mathbf{A}}(\mathcal{T})$, then $[\mathbf{A}_R]C \in closure_{\mathbf{A}}(\mathcal{T})$,
- if $[\mathsf{A}]C \in closure_{\mathbf{A}}(\mathcal{T})$ and $q \in Q_{\mathbf{A}}$, then $[\mathsf{A}_q]C \in closure_{\mathbf{A}}(\mathcal{T})$,
- if $[\mathsf{A}]_{\exists}C \in closure_{\mathbf{A}}(\mathcal{T})$ and $q \in Q_{\mathbf{A}}$, then $[\mathsf{A}_q]_{\exists}C \in closure_{\mathbf{A}}(\mathcal{T})$.

Observe that $closure_{\mathbf{A}}(\mathcal{T})$ is finite.

Lemma A.1. Algorithm 1 runs in polynomial time in the size of $\mathcal{A}$ (when assuming that $\mathbf{R}_+$, $\mathcal{R}$ and $\mathcal{T}$ are fixed).

Proof:
We will refer to the data structures used in Algorithm 1. Let n be the size of $\mathcal{A}$. Since $\mathcal{R}$ and $\mathcal{T}$ are fixed, the size of $closure_{\mathbf{A}}(\mathcal{T})$ is bounded by a constant. Observe that, for $x \in \Delta \setminus \Delta_0$, $Label(x) \subseteq closure_{\mathbf{A}}(\mathcal{T})$, and for $a \in \Delta_0$, $Label(a) \setminus \{A \mid A(a) \in \mathcal{A}\} \subseteq closure_{\mathbf{A}}(\mathcal{T})$. Hence the sizes of these two sets are also bounded by a constant. Since each $x \in \Delta \setminus \Delta_0$ has a unique $Label(x) \subseteq closure_{\mathbf{A}}(\mathcal{T})$, the set $\Delta \setminus \Delta_0$ contains only $O(1)$ elements. Hence, the size of Δ is of rank $O(n)$. Observe that:

- Function $\texttt{Find}(X)$ for $X \subseteq closure_{\mathbf{A}}(\mathcal{T})$ runs in constant time,
- Function $\texttt{CheckPremise}(x, C)$ for a fixed C runs in $O(n)$ steps,

- Procedure `ExtendLabel`(z, X) runs in $O(n)$ steps for $X \subseteq closure_{\mathbf{A}}(\mathcal{T})$,
- each iteration of the "while" loop in Algorithm 1 runs in $O(n^2)$ steps.

An iteration of the "while" loop in Algorithm 1 makes changes only when some of the following occur:

1. $Label(a)$ for some $a \in \Delta_0$ is extended by a subset of $closure_{\mathbf{A}}(\mathcal{T})$,
2. a new node x is added to Δ,
3. some $Next(x, \exists R.C)$ or $LeastSucc(x, R)$ is defined the first time,
4. some $Next(x, \exists R.C)$ or $LeastSucc(x, R)$ changes value from y to some $y_* \in \Delta \setminus \Delta_0$ with $Label(y) \subsetneq Label(y_*)$.

As the sizes of $closure_{\mathbf{A}}(\mathcal{T})$, $\Delta \setminus \Delta_0$ and $Label(y)$ for $y \in \Delta \setminus \Delta_0$ are bounded by a constant, the "while" loop in Algorithm 1 executes only $O(n)$ iterations. Therefore, the "while" loop in Algorithm 1 and hence the whole Algorithm 1 run in time $O(n^3)$. □

Lemma A.2. If Algorithm 1 returns $true$, then KB has a pseudo-model.

Proof:
Suppose Algorithm 1 returns $true$ for KB. We will refer to the data structures used by that run of Algorithm 1. A pseudo-model of KB will be constructed by starting from Δ_0, then unfolding the remaining part of the graph constructed by Algorithm 1, and then completing the interpretation of roles $R \in \mathbf{R}$. For that we define Δ' and $Edges'$ as counterparts of Δ and $Edges$, respectively, together with a mapping $f : \Delta' \to \Delta$ and a queue $unresolved$ of elements of Δ' as follows:

- $\Delta' := \Delta_0$;
- $FirmEdges := \{\langle a, r, b\rangle \mid r(a, b) \in \mathcal{A}\}$, $PseudoEdges := \emptyset$;
- for each $a \in \Delta_0$, $f(a) := a$;
- add the elements of Δ_0 to $unresolved$;
- while $unresolved$ is not empty:
  - extract an element u from $unresolved$;
  - for each $\exists R.C \in Label(f(u))$:
    - add a new element v to Δ' and $unresolved$;
    - $f(v) := Next(f(u), \exists R.C)$;
    - add $\langle u, R, v\rangle$ to $FirmEdges$;
  - for each $R \in \mathbf{R}$ such that $\exists R.\top \notin Label(f(u))$:
    - add a new element v to Δ' and $unresolved$;
    - $f(v) := LeastSucc(f(u), R)$;

$*$ add $\langle u, R, v\rangle$ to $PseudoEdges$;

- $Edges' := FirmEdges \cup PseudoEdges$.

The resulting data structures can be infinite. Let $\mathcal{I}$ be the pseudo-interpretation specified by:

- $\Delta^{\mathcal{I}} = \Delta'$;
- for each $A \in \mathbf{C}$, $A^{\mathcal{I}} = \{u \in \Delta' \mid A \in Label(f(u))\}$;
- for all $R \in \mathbf{R}$, $R^{\mathcal{I}_\exists}$ are the least relations satisfying the following conditions:
  - $(\overline{R}^{\mathcal{I}_\exists})^{-1} \subseteq R^{\mathcal{I}_\exists}$,
  - if $\langle u, R, v\rangle \in FirmEdges$, then $\langle u, v\rangle \in R^{\mathcal{I}_\exists}$,
  - for every word $S_1 \dots S_k$ accepted by $\mathbf{A}_R$, $S_1^{\mathcal{I}_\exists} \circ \dots \circ S_k^{\mathcal{I}_\exists} \subseteq R^{\mathcal{I}_\exists}$;
- for all $R \in \mathbf{R}$, $R^{\mathcal{I}_\forall}$ are the least relations satisfying the following conditions:
  - $R^{\mathcal{I}_\exists} \subseteq R^{\mathcal{I}_\forall}$ and $(\overline{R}^{\mathcal{I}_\forall})^{-1} \subseteq R^{\mathcal{I}_\forall}$,
  - if $\langle u, R, v\rangle \in PseudoEdges$, then $\langle u, v\rangle \in R^{\mathcal{I}_\forall}$,
  - for every word $S_1 \dots S_k$ accepted by $\mathbf{A}_R$, $S_1^{\mathcal{I}_\forall} \circ \dots \circ S_k^{\mathcal{I}_\forall} \subseteq R^{\mathcal{I}_\forall}$.

Observe that $\mathcal{I}$ is well-defined as it satisfies the condition that, for every $x \in \Delta^{\mathcal{I}}$ and every $r \in \mathbf{R}_+$, if $Y = \{y \mid \langle x, y\rangle \in r^{\mathcal{I}_\exists}\} \neq \emptyset$, then $\{y \mid \langle x, y\rangle \in r^{\mathcal{I}_\forall}\} = Y$. This is due to the use of $[\mathbf{A}_{\overline{R}}]_\exists \exists R.\top$ for the saturation operator, the expansion rules $(\forall_1)$–$(\forall_3)$ and the construction of the set $PseudoEdges$.

We show that $\mathcal{I}$ is a pseudo-model of KB. For this it suffices to prove that $u \in \varphi^{\mathcal{I}}$ for every $u \in \Delta'$ and every $\varphi \in Label(f(u))$ of the form A, $\exists R.A$, $\forall R.A$ or $C \sqsubseteq D$. We prove this by induction on the structure of φ. Let $u \in \Delta'$ and suppose $\varphi \in Label(f(u))$.

- Case $\varphi = A$ is trivial.
- Case $\varphi = \exists R.A$: Since $\varphi \in Label(f(u))$, there exists $v \in \Delta^{\mathcal{I}}$ such that $\langle u, v\rangle \in R^{\mathcal{I}_\exists}$ and $Next(f(u), \exists R.A) = f(v)$. We have that $A \in Label(f(v))$. Hence, $v \in A^{\mathcal{I}}$ and $u \in \varphi^{\mathcal{I}}$.
- Case $\varphi = \forall R.A$: Let v be any element of $\Delta^{\mathcal{I}}$ such that $\langle u, v\rangle \in R^{\mathcal{I}_\forall}$. We show that $v \in A^{\mathcal{I}}$. Since $\langle u, v\rangle \in R^{\mathcal{I}_\forall}$, there exist a word $S_1 \dots S_k$ accepted by $\mathbf{A}_R$ by a run $q_0 = q_{\mathsf{A}}, q_1, \dots, q_k$ and elements $u_0 = u, u_1, \dots, u_{k-1}, u_k = v$ such that, for every $1 \leq i \leq k$:
  - $u_{i-1} = u_i$ and S_i is of the form $(B?)$ with $B \in Label(f(u_{i-1}))$; or
  - $\langle u_{i-1}, u_i\rangle \in S_i^{\mathcal{I}_\forall}$ and ($\langle u_{i-1}, S_i, u_i\rangle \in Edges'$ or $\langle u_i, \overline{S_i}, u_{i-1}\rangle \in Edges'$).

  Let $\mathsf{A} = \mathbf{A}_R$. Since $\varphi \in Label(f(u))$ and $\varphi = \forall R.A$, by saturation, we have that $[\mathbf{A}_R]A \in Label(f(u))$, which means $[\mathsf{A}]A \in Label(f(u))$ and $[\mathsf{A}_{q_0}]A \in Label(f(u_0))$. For each i from 1 to k, due to the expansion rules $(\forall_1)$–$(\forall_5)$ and the saturation operator, it follows that $[\mathsf{A}_{q_i}]A \in Label(f(u_i))$. Since $q_k \in F_{\mathsf{A}}$ and $u_k = v$, it follows that $A \in Label(f(v))$. Hence, $v \in A^{\mathcal{I}}$.

- Case $\varphi = (C \sqsubseteq D)$ and $C = C_1 \sqcap \ldots \sqcap C_k$: Suppose $u \in C^{\mathcal{I}}$. We prove that $u \in D^{\mathcal{I}}$. The last call $\texttt{CheckPremise}(f(u), C)$ returned *true* because the following observations hold for every $1 \leq i \leq k$:
  - Case $C_i = A$: Since $u \in C_i^{\mathcal{I}}$, we have that $A \in \mathit{Label}(f(u))$.
  - Case $C_i = \exists R.A$: Since $u \in C_i^{\mathcal{I}}$, there exist a word $S_1 \ldots S_h$ accepted by $\mathbf{A}_R$ and elements $u_0 = u, u_1, \ldots, u_{h-1}, u_h$ such that $u_h \in A^{\mathcal{I}}$ and, for every $1 \leq j \leq h$:
    - $u_{j-1} = u_j$ and S_j is of the form $(B?)$ with $B \in \mathit{Label}(f(u_{j-1}))$; or
    - $\langle u_{j-1}, u_j \rangle \in S_j^{\mathcal{I}_\exists}$ and ($\langle u_{j-1}, S_j, u_j \rangle \in \mathit{FirmEdges}$ or $\langle u_j, \overline{S_j}, u_{j-1} \rangle \in \mathit{FirmEdges}$).

    The word $\overline{S}_h \ldots \overline{S}_1$ is accepted by $\mathsf{A} = \mathbf{A}_{\overline{R}}$ by a run $q_h = q_{\mathsf{A}}, q_{h-1}, \ldots, q_0$. Since $u_h \in A^{\mathcal{I}}$, we have that $A \in \mathit{Label}(f(u_h))$ and, by saturation, $[\mathbf{A}_{\overline{R}}]_\exists \langle \mathbf{A}_R \rangle A \in \mathit{Label}(f(u_h))$, which means $[\mathsf{A}_{q_h}]_\exists \langle \mathbf{A}_R \rangle A \in \mathit{Label}(f(u_h))$. For each j from h to 1, due to the expansion rules $(\forall_1)$–$(\forall_3)$ and the saturation operator, it follows that $[\mathsf{A}_{q_{j-1}}]_\exists \langle \mathbf{A}_R \rangle A \in \mathit{Label}(f(u_{j-1}))$. Since $q_0 \in F_{\mathsf{A}}$ and $u_0 = u$, it follows that $\langle \mathbf{A}_R \rangle A \in \mathit{Label}(f(u))$.
  - Case $C_i = \forall \exists R.A$ with $R \in \mathbf{R}$: Since $u \in C_i^{\mathcal{I}}$, we have that $u \in (\exists R.\top)^{\mathcal{I}}$. Thus, there exist a word $S_1 \ldots S_h$ accepted by $\mathbf{A}_R$ and elements $u_0 = u, u_1, \ldots, u_{h-1}, u_h$ such that, for every $1 \leq j \leq h$, $\langle u_{j-1}, u_j \rangle \in S_j^{\mathcal{I}_\exists}$, and $\langle u_{j-1}, S_j, u_j \rangle \in \mathit{FirmEdges}$ or $\langle u_j, \overline{S_j}, u_{j-1} \rangle \in \mathit{FirmEdges}$. The word $\overline{S}_h \ldots \overline{S}_1$ is accepted by $\mathsf{A} = \mathbf{A}_{\overline{R}}$ by a run $q_h = q_{\mathsf{A}}, q_{h-1}, \ldots, q_0$. By saturation, $[\mathbf{A}_{\overline{R}}]_\exists \exists R.\top \in \mathit{Label}(f(u_h))$, which means $[\mathsf{A}_{q_h}]_\exists \exists R.\top \in \mathit{Label}(f(u_h))$. For each j from h to 1, due to the expansion rules $(\forall_1)$–$(\forall_3)$, it follows that $[\mathsf{A}_{q_{j-1}}]_\exists \exists R.\top \in \mathit{Label}(f(u_{j-1}))$. Since $q_0 \in F_{\mathsf{A}}$ and $u_0 = u$, it follows that $\exists R.\top \in \mathit{Label}(f(u))$. There exists $v \in \Delta^{\mathcal{I}}$ with $f(v) = \mathit{Next}(f(u), \exists R.\top)$. We have that $\langle u, v \rangle \in R^{\mathcal{I}_\exists}$. Since $u \in (\forall \exists R.A)^{\mathcal{I}}$, it follows that $v \in A^{\mathcal{I}}$ and hence $A \in \mathit{Label}(f(v))$, which means $A \in \mathit{Label}(\mathit{Next}(f(u), \exists R.\top))$.
  - Case $C_i = \forall R.A$: Observe that, for any $v \in \Delta'$, any finite automaton A over Σ and any $B \in \mathbf{C}$, if $(G, f(v) \not\models_c [\mathsf{A}]B)$, then $v \notin ([\mathsf{A}]B)^{\mathcal{I}}$. This can be proved by induction on the construction of the relation $\not\models_c$. Since $u \in C_i^{\mathcal{I}}$, we have that $u \in ([\mathbf{A}_R]A)^{\mathcal{I}}$, which implies $G, f(u) \models_c [\mathbf{A}_R]A$ and $G, f(u) \models_c \forall R.A$.

  We have shown that $\texttt{CheckPremise}(f(u), C)$ returned *true*. It follows that $D \in \mathit{Label}(f(u))$, and by the inductive assumption, $u \in D^{\mathcal{I}}$.

  □

Given an interpretation $\mathcal{I}$, for $\varphi = (C \sqsubseteq D)$, define $\varphi^{\mathcal{I}} = (\neg C \sqcup D)^{\mathcal{I}}$, and for a set X consisting of concepts and TBox axioms, define $X^{\mathcal{I}} = \bigcap\{\varphi^{\mathcal{I}} \mid \varphi \in X\}$.

As Algorithm 1 tries to derive $\bot$ at some node of the constructed graph, Lemma A.2 given above is in fact an assertion about the completeness of the procedure. It remains to show the soundness: if the rule $(\bot)$ is applicable, then KB does not have any pseudo-model. It is sufficient to show that every change made to the graph constructed by Algorithm 1 is "justifiable". An informal justification for this has been given in the discussion about the algorithm. For a formal justification, we consider the contrapositive assertion: if KB has a pseudo-model, then Algorithm 1 returns *true* for it. By assuming that KB has a pseudo-model $\mathcal{I}$, every change made to the constructed graph can be justified by $\mathcal{I}$. In particular, $\bot$ cannot be added to the label of any node of the graph. This is formalized by the following lemma.

Lemma A.3. Suppose that a clausal Horn-CPDL$_{reg}$ knowledge base $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ has a pseudo-model $\mathcal{I}$. Consider an execution of Algorithm 1 for KB and any moment after executing the step 6 of that execution. Let $Z = \{\langle x, u \rangle \in \Delta \times \Delta^{\mathcal{I}} \mid u \in (Label(x))^{\mathcal{I}}\}$. Then:

1. for every $a \in \mathbf{I}$ occurring in $\mathcal{A}$, $Z(a, a^{\mathcal{I}})$ holds;
2. for every $x, y \in \Delta$, $u, v \in \Delta^{\mathcal{I}}$ and $\exists R.A$ such that $Next(x, \exists R.A) = y$, if $Z(x, u)$, $R^{\mathcal{I}_\exists}(u, v)$ and $v \in A^{\mathcal{I}}$ hold, then $Z(y, v)$ holds;
3. for every $x, y \in \Delta$, $u, v \in \Delta^{\mathcal{I}}$ and R such that $Next(x, \exists R.\top) = y$, if $Z(x, u)$ and $R^{\mathcal{I}_\exists}(u, v)$ hold, then $Z(y, v)$ holds;
4. for every $x, y \in \Delta$, $u, v \in \Delta^{\mathcal{I}}$ and R such that $LeastSucc(x, R) = y$, if $Z(x, u)$ and $R^{\mathcal{I}_\forall}(u, v)$ hold, then $Z(y, v)$ holds;
5. for every $x \in \Delta$, there exists $u \in \Delta^{\mathcal{I}}$ such that $Z(x, u)$ holds. □

Note that, if $Z(x, u)$ holds, then $u \in (Label(x))^{\mathcal{I}}$, which means $Label(x)$ is satisfied at (and hence "justified by") u in $\mathcal{I}$. The first four assertions of this lemma can be proved by induction on the number of executed steps in a way similar to the proof of [20, Lemma 3.5]. The last assertion follows from the previous assertions, because every $x \in \Delta \setminus \Delta_0$ is/was at some step reachable from Δ_0 and $Label(x)$ was never changed.

Corollary A.4. If a clausal Horn-CPDL$_{reg}$ knowledge base KB has a pseudo-model, then Algorithm 1 returns $true$ for it.

Proof:
By the last assertion of Lemma A.3, the rule $(\bot)$ was never applicable. This means that Algorithm 1 does not return $false$. As it always terminates (by Lemma A.1), it must return $true$. □

Lemma A.5. Let $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ be a clausal Horn-CPDL$_{reg}$ knowledge base specified without using the constructor $\forall R_{l\forall}.C_l$ in the grammar rule (8). If KB has a pseudo-model, then it also has a model.

Proof:
Suppose KB has a pseudo-model and consider a run of Algorithm 1 for KB. By Corollary A.4, it returns *true*. Let the sets Δ', $FirmEdges$, $PseudoEdges$ and the mapping $f : \Delta' \to \Delta$ be constructed as in the proof of Lemma A.2. Let $\mathcal{I}$ be the interpretation specified by:

- $\Delta^{\mathcal{I}} = \Delta'$,
- for each $A \in \mathbf{C}$, $A^{\mathcal{I}} = \{u \in \Delta' \mid A \in Label(f(u))\}$,
- for all $R \in \mathbf{R}$, $R^{\mathcal{I}}$ are the least relations satisfying the following conditions:
  - $(\overline{R}^{\mathcal{I}})^{-1} \subseteq R^{\mathcal{I}}$,
  - if $\langle u, R, v \rangle \in FirmEdges$, then $\langle u, v \rangle \in R^{\mathcal{I}}$,
  - for every word $S_1 \dots S_k$ accepted by $\mathbf{A}_R$, $S_1^{\mathcal{I}} \circ \dots \circ S_k^{\mathcal{I}} \subseteq R^{\mathcal{I}}$.

Similar to the proof of Lemma A.2, it can be proved that $\mathcal{I}$ is a model of KB. □