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 ∀R.C and ∃R.C correspond to the modalities [R]C and ⟨R⟩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 ALC is NP-hard [7]. The hardness is caused by basic roles not being required to be serial (i.e., to satisfy the condition ∀x∃y R(x, y)). A naive approach for overcoming the NP-hardness is to disallow the concept constructor ∀R.C on the LHS (left hand side) of ⊑ in TBox axioms [8, 9, 10, 11, 12, 13, 14, 15, 16].

EL [9, 10], DL-Lite [17, 16], DLP [8], Horn-SHIQ [11] and Horn-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 ∀R.C on the LHS of ⊑ in TBox axioms.

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

- allowing ∀R.C to appear on the LHS of ⊑ in TBox axioms when R is serial and using the traditional semantics for it,
- allowing a special kind of ∀R.C like ∀∃R.C (defined as ∀R.C ∩ ∃R.C) to appear on the LHS of ⊑ in TBox axioms and using the traditional semantics for it,
- allowing ∀R.C to appear on the LHS of ⊑ 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 → in program clauses (like allowing ∀R.C to appear on the LHS of ⊑ 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 \( \sigma \), the combined modal operator \( [G_{\sigma}]_{\circ} \phi \) defined as \( [G_{\sigma}]_{\circ} \phi \land (G_{\sigma})_{\circ} \phi \) is allowed instead of the universal modal operator \( [G_{\sigma}]_{\circ} \phi \) 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 \( [\pi]_{\circ} \phi \) to appear on the LHS of \( \rightarrow \) in program clauses. This modal operator is stronger than \( [\pi] \phi \land (\pi) \phi \). The formula \( [\pi]_{\circ} \phi \) means that \( [\pi] \phi \) holds and every partial run of \( \pi \) (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 \( \subseteq \) in TBox axioms was introduced by us in [24]. It is the deterministic Horn fragment of ALC, for which the constructor \( \forall R.C \), defined as \( \forall R.C \land \exists R.C \) [19], is allowed on the LHS of \( \subseteq \) 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 Reg, which we denote by Horn-Reg. 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-Reg, Horn-SHIQ and Horn-SROIQ, while still having PTIME data complexity. In comparison with Horn-Reg [7], Horn-DL additionally allows inverse roles, nominals, qualified number restrictions, the \( \exists r.Self \) constructor, the universal role as well as assertions of the form disjoint(\( s, s' \)), irreflexive(\( s \)), \( \neg s(a, b) \), \( a \neq b \). In comparison with Horn-SROIQ, Horn-DL additionally allows the universal role and assertions of the form irreflexive(\( s \)), \( \neg s(a, b) \), \( a \neq b \). More importantly, in contrast to all the well-known Horn fragments EL [9, 10], DL-Lite [17], DLP [8], Horn-SHIQ [11], Horn-SROIQ [11] of DLs, Horn-DL allows the concept constructor \( \forall \exists R.C \) to appear on the LHS of \( \subseteq \) 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 \( R \in R \), the concept constructor \( \forall \exists R.C \) is more constructive than \( \forall R.C \) on the LHS of \( \subseteq \) in TBox axioms. For the case when \( R \) is not a basic role, a constructor similar to \( [\pi]_{\circ} \phi \) 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 \( \subseteq \) 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 \( \subseteq \) 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 $\text{PTIME}$ 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 $\text{PTIME}$ 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 $C$ of concept names, a finite set $R_+$ of role names, and a finite set $I$ 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 $\tau$ to denote the inverse of $r$. For $R = \tau$, let $R$ stand for $r$. Let $R_- = \{ r \mid r \in R_+ \}$ and $R = R_+ \cup R_-$. We call the roles from $R$ basic roles.

2.1. Regular RBoxes

A context-free semi-Thue system $S$ over $R$ is a finite set of context-free production rules $R \rightarrow S_1 \ldots S_k$ over alphabet $R$ (i.e., $R, S_1, \ldots, S_k \in R$). It is symmetric if, for every rule $R \rightarrow S_1 \ldots S_k$ of $S$, the rule $\overline{R} \rightarrow \overline{S_k} \ldots \overline{S_1}$ is also in $S$. It is regular if, for every $R \in R$, the set of words derivable from $R$ using the system is a regular language over $R$.

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 $R \in R$, the word $R$ is derivable from $R$ using such a system.

A role inclusion axiom (RIA for short) is an expression of the form $S_1 \circ \cdots \circ S_k \subseteq R$, where $k \geq 0$ and $S_1, \ldots, S_k, R \in R$. In the case $k = 0$, the LHS of the inclusion axiom stands for the empty word $\varepsilon$.

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

$$\{ R \rightarrow S_1 \ldots S_k \mid (S_1 \circ \cdots \circ S_k \subseteq R) \in \mathcal{R} \}$$

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

Recall that a finite automaton $A$ over alphabet $R$ is a tuple $\langle R, Q, q_0, \delta, F \rangle$, where $Q$ is a finite set of states, $q_0 \in Q$ is the initial state, $\delta \subseteq Q \times R \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. A run of $A$ on a word $R_1 \ldots R_k$ over alphabet $R$ is a finite sequence of states $q_0, q_1, \ldots, q_k$ 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 $L(A)$.

1 In the case $k = 0$, the right hand sides of the rules stand for $\varepsilon$. 
Example 2.1. Let $R_+=\{r\}$ and $R=\{\tau \circ r \subseteq r, \tau \circ r \subseteq \tau\}$. The symmetric regular semi-Thue system corresponding to $R$ is $S=\{r \rightarrow \tau r, \tau \rightarrow \tau r\}$. The set of words derivable from $r$ (resp. $\tau$) using $S$ is a regular language characterized by the regular expression $r \cup (\tau; (\tau \cup r)^*; r)$ (resp. $\tau \cup (\tau; (\tau \cup r)^*; r)$). Hence, $R$ is a regular RBox, whose RIA-automaton-specification $A$ is specified by:

$$A_r = (R, \langle 0, 1, 2 \rangle, 0, \{\langle 0, r, 1 \rangle, \langle 0, \tau, 2 \rangle, \langle 2, r, 2 \rangle, \langle 2, \tau, 2 \rangle, \langle 2, r, 1 \rangle\}, \{1\})$$

$$A_\tau = (R, \langle 0, 1, 2 \rangle, 0, \{\langle 0, \tau, 1 \rangle, \langle 0, r, 2 \rangle, \langle 2, r, 2 \rangle, \langle 2, r, 2 \rangle, \langle 2, r, 1 \rangle\}, \{1\})$$

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

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

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

Concepts and roles are defined, respectively, by the following BNF grammar rules, where $A \in C$ and $r \in R_+$:

$$C ::= \top | \bot | A | \neg C | C \cap C | C \cup C | \forall R.C | \exists R.C$$

$$R ::= \varepsilon | r | R \circ R | R \triangleright R | R^* | C?$$

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$_{\text{reg}}$ is a tuple $(R, T, A)$ consisting of a regular RBox $R$, a TBox $T$ and an ABox $A$.

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

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

$$I \models S_1 \circ \cdots \circ S_k \subseteq R \quad \text{if} \quad S_1^I \circ \cdots \circ S_k^I \subseteq R^I$$

$$I \models \varepsilon \subseteq R \quad \text{if} \quad \varepsilon^I \subseteq R^I$$

$$I \models C \subseteq D \quad \text{if} \quad C^I \subseteq D^I$$

$$I \models C(a) \quad \text{if} \quad a^I \in C^I$$

$$I \models r(a, b) \quad \text{if} \quad \langle a^I, b^I \rangle \in r^I.$$  

If $I \models \varphi$, then we say that $I$ validates $\varphi$. 

\[\top^I = \Delta^I \quad \bot^I = \emptyset\]
\[\neg C^I = \Delta^I \setminus C^I\]
\[(C \cap D)^I = C^I \cap D^I\]
\[(C \cup D)^I = C^I \cup D^I\]
\[(\forall R.C)^I = \{x \in \Delta^I \mid \forall y ((x, y) \in R^I \Rightarrow y \in C^I)\}\]
\[(\exists R.C)^I = \{x \in \Delta^I \mid \exists y ((x, y) \in R^I \land y \in C^I)\}\]

\[\varepsilon^I = \{(x, x) \mid x \in \Delta^I\}\]
\[\mathcal{R}^I = (R^I)^{-1}\]
\[(R \circ S)^I = R^I \circ S^I\]
\[(R \cup S)^I = R^I \cup S^I\]
\[(R^*)^I = (R^I)^*\]
\[(C^?)^I = \{(x, x) \mid x \in C^I\}\]

\[\forall(R.A)^I = \{\text{true} \mid \forall(y, x) \in R^I \Rightarrow y \in A^I\}\]
\[\forall(R.A)^I = \{\text{true} \mid \forall(y, x) \in R^I \land y \in A^I\}\]

\[\forall(R.A)^I = \{(x, x) \mid x \in A^I\}\]

\[\forall(R.A)^I = \{(x, x) \mid x \in A^I\}\]

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 \(\varphi\). If \(KB \models C(a)\), then we say that \(a\) is an instance of \(C\) w.r.t. \(KB\).

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

\[\top \subseteq \forall \text{hasSon}.\text{Male},\]
\[\top \subseteq \forall \text{hasDaughter}.\text{Female},\]
\[\text{Male} \cap \text{Female} \subseteq \bot,\]
\[A \subseteq \forall \text{hasSon}.B \cap \forall \text{hasDaughter}.C \cap D,\]
\[\text{Male} \cap B \subseteq \forall (\text{hasSon} \cup \text{hasDaughter})^*.D,\]
\[\text{Female} \cap C \subseteq \forall (\text{hasSon} \cup \text{hasDaughter})^*.D,\]
\[\forall (\text{hasSon} \cup \text{hasDaughter})^*.D \subseteq E.\]

Let \(KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle\) and consider the interpretation \(\mathcal{I}\) specified by:
\[\Delta^I = \{a, b, c\}, \text{Lily}^I = a, \text{Jack}^I = b,\]
\[\text{hasSon}^I = \{(a, b), (b, c)\}, \text{hasDaughter}^I = \emptyset,\]
\[\text{Female}^I = \{a\}, \text{Male}^I = \{b, c\},\]
\[A^I = \{a\}, B^I = \{b\}, C^I = \emptyset, D^I = E^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., \(\mathcal{I} \models (\forall \text{hasDaughter}.\text{Male})(Lily)\) but \(KB \not\models (\forall \text{hasDaughter}.\text{Male})(Lily)\)). Also observe that \(KB \models E(Lily)\). \(\square\)
The size of a concept, a role, an assertion or an axiom $\varphi$ is defined to be the number of bits needed to encode $\varphi$ 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 $\langle R, T, A \rangle \models C(a)$ is defined when $A$ is a reduced ABox and is measured w.r.t. the size of $A$, while assuming that $R_+, R, T$ and $C(a)$ are fixed.

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

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

$$
C_l ::= T | A | C_l \cap C_l | C_l \cup C_l | \exists R_{l\exists} C_l | \forall R_{l\forall} C_l | \forall \exists r.C_l | \forall \exists \forall C_l 
$$

(8)

$$
R_{l\exists} ::= r | \overline{R_{l\exists}} | R_{l\exists} \circ R_{l\exists} | R_{l\exists} \sqcup R_{l\exists} | R_{l\exists}^* | C_l?
$$

(9)

$$
R_{l\forall} ::= r | \overline{R_{l\forall}} | R_{l\forall} \circ R_{l\forall} | R_{l\forall} \sqcup R_{l\forall} | R_{l\forall}^* | (\neg C_l)?
$$

(10)

$$
C_r ::= T | \bot | A | \neg C_l | C_r \cap C_r | \neg C_l \cup C_r | \exists R_{r\exists} C_r | \forall R_{r\forall} C_r
$$

(11)

$$
R_{r\exists} ::= r | \overline{R_{r\exists}} | R_{r\exists} \circ R_{r\exists} | C_r?
$$

(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 $C_l \cap \ldots \cap C_k \subseteq D$ or $\top \subseteq D$, where:

- each $C_l$ is of the form $A, \exists R_{l\exists} A, \forall R_{l\forall} A, \forall r. A$ or $\forall \exists \forall A$,
- $D$ is of the form $\bot, \exists r. T, \exists \forall T, A, \exists r. A, \exists \forall A$ or $\forall R_{r\exists} A$,
- $R_{l\exists}$ and $R_{l\forall}$ are now restricted by the following BNF grammar:

$$
R_{l\exists} ::= r | \overline{R_{l\exists}} | R_{l\exists} \circ R_{l\exists} | R_{l\exists} \sqcup R_{l\exists} | R_{l\exists}^* | A?
$$

(13)

$$
R_{l\forall} ::= r | \overline{R_{l\forall}} | R_{l\forall} \circ R_{l\forall} | R_{l\forall} \sqcup R_{l\forall} | R_{l\forall}^* | (\neg A)\
$$

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 $\langle R, T, A \rangle$ consisting of a regular RBox $R$, a Horn-CPDL$_{reg}$ TBox $T$ and a Horn-CPDL$_{reg}$ ABox $A$. When $T$ is a clausal Horn-CPDL$_{reg}$ TBox and $A$ 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 $a \in I$ 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.

A clause of the form $C \subseteq \exists R. \top$ can be replaced by $C \subseteq \exists R.A_\top$, where $A_\top$ is a fresh concept name. We allow such clauses just for convenience.
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 $(\exists \text{hasSon.} \top)(\text{Jack})$ by $F(\text{Jack})$, where $F$ is a new concept name (and now $C = \{\text{Male, Female, A, B, C, D, E, F}\}$);
- $\mathcal{T}_2$ is obtained from $\mathcal{T}$ by adding the axiom $F \sqsubseteq \exists \text{hasSon.} \top$ and replacing the axiom (4) by $(A \sqsubseteq \forall \text{hasDaughter.} C)$ and $(A \sqsubseteq D)$. 

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

The objective is to define an appropriate satisfaction relation $KB \models C(a)$ for a Horn-CPDL$_{reg}$ knowledge base $KB$ and a Horn-CPDL$_{reg}$ query $C(a)$. The intention is that $KB \models C(a)$ 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 $KB \models C(a)$, then $KB \models C(a)$. That is, $\models$ is sound w.r.t. the traditional semantics.
- If $KB$ and $C$ are specified without using the constructor $\forall R_{ty}, C_i$ in the grammar rule (8), then $KB \models C(a)$ iff $KB \models C(a)$.
- If $\{T \sqsubseteq \exists R, T \mid R \in \mathcal{R}\} \subseteq \mathcal{T}$ (i.e., all $R \in \mathcal{R}$ are serial), then $KB \models C(a)$ 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 $\mathcal{I} = \langle \Delta^I, \cdot^I \rangle$, where $\Delta^I$ is a non-empty set called the domain of $\mathcal{I}$ and $\cdot^I$ is a mapping called the interpretation function of $\mathcal{I}$ that associates each individual name $a \in I$ with an element $a^I \in \Delta^I$, each concept name $A \in C$ with a set $A^I \subseteq \Delta^I$, and each role name $r \in \mathcal{R}_+$ with a pair $(r^I, r^I_v)$ of binary relations such that:
• \( r^I_a \subseteq r^I \subseteq \Delta^I \times \Delta^I \),
• for every \( x \in \Delta^I \), if \( Y = \{ y \mid \langle x, y \rangle \in r^I_a \} \neq \emptyset \), then \( \{ y \mid \langle x, y \rangle \in r^I \} = Y \).

The interpretation function \( \mathcal{I} \) 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 \( R^I_a \subseteq R^I \), and \( (\forall R.C)^I \) may differ from \( (\neg \exists R.\neg C)^I \). If \( \langle x, y \rangle \in R^I_a \), then we call \( \langle x, y \rangle \) a firm \( R \)-edge. If \( \langle x, y \rangle \in R^I \setminus R^I_a \), then we call \( \langle x, y \rangle \) a pseudo \( R \)-edge.

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

\[
\begin{align*}
\mathcal{I} \models S_1 \circ \cdots \circ S_k & \subseteq R & \text{if} & & S_1^I \circ \cdots \circ S_k^I \subseteq R^I_a \quad \text{and} \quad S_1^I \circ \cdots \circ S_k^I \subseteq R^I \\
\mathcal{I} \models \varepsilon & \subseteq R & \text{if} & & \varepsilon^I \subseteq R^I_a \\
\mathcal{I} \models C & \subseteq D & \text{if} & & C^I \subseteq D^I \\
\mathcal{I} \models C(a) & \text{if} & a^I \in C^I \\
\mathcal{I} \models r(a, b) & \text{if} & \langle a^I, b^I \rangle \in r^I_a.
\end{align*}
\]

If \( \mathcal{I} \models \varphi \), then we say that \( \mathcal{I} \) validates \( \varphi \). 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 \( \mathcal{I} \models KB \), 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  \( \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle \models C(a) \) if, for every pseudo-model \( \mathcal{I} \) of \( \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle \), it holds that \( \mathcal{I} \models C(a) \).  

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

\[
\begin{align*}
A \sqcap \forall r.B & \sqsubseteq C, \\
A \sqcap \exists r.\tau & \sqsubseteq D, \\
C \sqcup D & \sqsubseteq E.
\end{align*}
\]

Then \( KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle \) is a Horn-CPDL\textsubscript{reg} knowledge base. Observe that \( KB \nvdash C(a) \) and \( KB \nvdash D(a) \), 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:

\[
\begin{array}{c}
\text{a} : A \\
\text{b} : \text{a}
\end{array}
\]

\[\Delta^I = \{ a, b \}, a^I = a, A^I = \{ a \}, B^I = C^I = D^I = E^I = \emptyset, r^I_a = \emptyset, r^I = \{ \langle a, b \rangle, \langle b, b \rangle \} \]. Observe that \( \mathcal{I} \models KB \) and \( \mathcal{I} \nvdash E(a) \). This shows that the constructive semantics is “weaker” (i.e., allows to derive fewer logical consequences) than the traditional semantics. 

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

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

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

Remark 4.5. An interpretation $I$ can be treated as a pseudo-interpretation with $r^I_\exists = r^I_\forall = r^I$ for all $r \in R_+$. Thus, given an interpretation $I$, $I \models KB$ iff $I \models KB_2$, and $I \models C(a)$ iff $I \models C(a)$. Conversely, a pseudo-interpretation $I$ satisfying $r^I_\exists = r^I_\forall = r^I$ for all $r \in R_+$ can be treated as an interpretation. In particular, if $I \models (\top \sqsubseteq \exists R. \top)$ for all $R \in R$, then $I$ 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 $R^I = S^I$ in every interpretation $I$). For example, $(r_1 \sqcup r_2) \sqcap r_3^I \equiv r_3^I \sqcap (r_1 \sqcup r_2)$.

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

1. If $C(a)$ is a Horn-$\text{CPDL}_{reg}$ query, then, for the Horn-$\text{CPDL}_{reg}$ knowledge base $KB' = \langle R, \mathcal{T} \cup \{C \sqsubseteq A\}, A \cup \{\neg A(a)\} \rangle$, where $A$ is a fresh concept name, we have that:
   (a) $KB \models C(a)$ 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 $T$ and $A$ to a Horn-CPDL$_{reg}$ knowledge base $KB' = \langle R, T', A' \rangle$ with $A'$ 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 $T$ to a Horn-CPDL$_{reg}$ knowledge base $KB' = \langle R, T', A \rangle$ with $T'$ 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 $T$ does not use the constructor $\forall R.C$ on the LHS of $\sqsubseteq$, then $T'$ does neither.

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

For the third assertion, we first transform roles occurring in $T$ 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 $T$, 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 $T$ and $A$ to a clausal Horn-CPDL$_{reg}$ knowledge base $KB' = \langle R, T', A' \rangle$ 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 $KB \models C(a)$, then $KB \models C(a)$.
2. If $\{ \top \sqsubseteq \exists R.\top \mid R \in R \} \subseteq T$, then:
(a) if \( KB \) has a pseudo-model, then it also has a model,
(b) \( KB \models C(a) \) iff \( KB \models C(a) \).

3. If \( KB \) is specified without using the constructor \( \forall R_{lv}.C_l \) 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 \( \forall R_{lv}.C_l \) in the grammar rule (8), then \( KB \models C(a) \) iff \( KB \models C(a) \). \( \square \)

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\(_{\text{reg}}\)

In this section we present an algorithm that, given a clausal Horn-CPDL\(_{\text{reg}}\) knowledge base \( KB = \langle R, T, A \rangle \) together with the RIA-automaton-specification \( A \) of \( R \), 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 \( \Sigma = R \cup \{ A?, (\neg A)? \mid A \in C \} \) (where \( \circ \) corresponds to \( ; \) and \( \cup \) corresponds to \( \cup \)). The regular language characterized by such a role \( R \) is denoted by \( L(R) \). A word \( R_1 R_2 \ldots R_k \) over \( \Sigma \) is also treated as the role \( R_1 \circ R_2 \circ \cdots \circ R_k \).

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

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

Given a role \( R \) in ITNF, by \( A_{\overline{R}} \) we denote \( A_S \) 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) \( I \) and a finite automaton \( A \) over alphabet \( \Sigma \), we define \( A^I \) (resp. \( A^{\overline{I}}, A^{\Sigma} \)) to be \( \{ (x, y) \in \Delta^I \times \Delta^I \mid \text{there exist a word } R_1 \ldots R_k \text{ accepted by } A \text{ and elements } x_0 = x, x_1, \ldots, x_k = y \text{ of } \Delta^I \text{ such that } \langle x_{i-1}, x_i \rangle \in R_i^I, \langle x_{i-1}, x_i \rangle \in R_i^{\Sigma}, \langle x_{i-1}, x_i \rangle \in R_i^{\overline{I}} \text{ for all } 1 \leq i \leq k \}. \)

We will use auxiliary concept constructors \( [A]C, [A]_\exists C \) and \( \langle A \rangle C \), where \( A \) is a finite automaton over alphabet \( \Sigma \) 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 \( [A]C, [A]_\exists C, \langle A \rangle C \) are specified below:
• given an interpretation \( I \),

\[
([A]C)^I = \{ x \in \Delta^I \mid \forall y ((x, y) \in A^I \implies y \in C^I) \},
\]
\[
((A)C)^I = \{ x \in \Delta^I \mid \exists y ((x, y) \in A^I \land y \in C^I) \};
\]

• given a pseudo-interpretation \( I \),

\[
([A]C)^I = \{ x \in \Delta^I \mid \forall y ((x, y) \in A^I \implies y \in C^I) \},
\]
\[
([A]∃C)^I = \{ x \in \Delta^I \mid \exists y ((x, y) \in A^I \exists \implies y \in C^I) \},
\]
\[
((A)C)^I = \{ x \in \Delta^I \mid \exists y ((x, y) \in A^I \land y \in C^I) \};
\]

For a finite automaton \( A \) over \( \Sigma \), let the components of \( A \) be denoted as in

\[
A = \langle \Sigma, Q, q_0, \delta, F \rangle.
\]

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 \( I \) be a pseudo-model of a regular RBox \( R \), \( A \) the RIA-automaton-specification of \( R \), and \( C \) a concept. Then:

• \( (\forall R.C)^I = ([A]R[C])^I \) and \( (\exists R.C)^I = ((A)_R C)^I \),

\[ C^I \subseteq ([A]_{\neg R} [A] R C)^I \] and \( C^I \subseteq ([A]_{\neg R} \exists R.C)^I \).

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. \( A \) and \( T \)), denoted by \( \text{Satr}(X) \), is defined to be the least extension of \( X \) such that:

1. for every \( R \in \mathbf{R} \), \([A]_{\neg R} \exists R.C \in \text{Satr}(X)\),

2. if \( \forall R.C \in \text{Satr}(X) \), then \([A]_{R} C \in \text{Satr}(X)\),

3. if \([A]C \in \text{Satr}(X)\), \((q_A, B?, q) \in \delta_A \) and \( B \in \text{Satr}(X) \), then \([A]_q C \in \text{Satr}(X)\),

4. if \([A]_3 C \in \text{Satr}(X)\), \((q_A, B?, q) \in \delta_A \) and \( B \in \text{Satr}(X) \), then \([A]_{q_A} C \in \text{Satr}(X)\),

5. if \((A)C \in \text{Satr}(X) \) or \([A]_3 C \in \text{Satr}(X)) \) and \( q_A \in F_A \), then \( C \in \text{Satr}(X)\),

6. if \( B \in \text{Satr}(X) \) and \( \exists R.B \) occurs on the LHS of \( \sqcup \) in some clause of \( T \), then \([A]_{\neg R} [A] R B \in \text{Satr}(X)\).\(^3\)

\(^3\)As stated in Section 2, the letter \( B \) denotes a concept name (i.e., an atomic concept).
For $R \in \mathbb{R}$, there are two kinds of transfer of $X$ through $R$:

$$\text{Trans}(X, R) = \{ [[A_q]C \mid [A]C \in X \text{ and } \langle q_A, R, q \rangle \in \delta_A \}$$

$$\text{Trans}_\varnothing(X, R) = \text{Trans}(X, R) \cup \{ [[A]_\varnothing \mid [A]_\varnothing \in X \text{ and } \langle q_A, R, q \rangle \in \delta_A \}.$$

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

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

Define $\text{Edges} = \{ \langle x, R, y \rangle \mid R(x, y) \in A \text{ or Next}(x, \exists R. C) = y \text{ for some } C \text{ or } \text{LeastSucc}(x, R) = y \}$. A tuple $\langle x, R, y \rangle \in \text{Edges}$ represents an edge $\langle x, y \rangle$ labeled by $R$ of the graph. If $R(x, y) \in A$ or $\text{Next}(x, \exists R. C) = y$, then we call $\langle x, R, y \rangle$ a firm edge, else if $\text{LeastSucc}(x, R) = y$, 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 $\Delta_0$ if there exist $x_0, \ldots, x_k \in \Delta$ and elements $R_1, \ldots, R_k$ of $R$ such that $k \geq 0$, $x_0 \in \Delta_0$, $x_k = x$ and $\langle x_{i-1}, R_i, x_i \rangle \in \text{Edges}$ for all $1 \leq i \leq k$.

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

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

- $q_A \in F_A$ and $B \notin \text{Label}(x)$;
- $\langle q_A, (\neg B')?, q \rangle \in \delta_A$, $B' \notin \text{Label}(x)$ and $G, x \not\models_c [A_q]B$;
- $\langle q_A, R, q \rangle \in \delta_A$, $\exists R. T \notin \text{Label}(x)$ and $\text{LeastSucc}(x, R)$ is not defined;
- $\langle q_A, R, q \rangle \in \delta_A$, $\exists R. T \notin \text{Label}(x)$, $\text{LeastSucc}(x, R) = y$ and $G, y \not\models_c [A_q]B$;
- $\langle q_A, R, q \rangle \in \delta_A$, $\exists R. T \in \text{Label}(x)$ and $\text{Next}(x, \exists R. T)$ is not defined;
- $\langle q_A, R, q \rangle \in \delta_A$, $\text{Next}(x, \exists R. T) = y$ and $G, y \not\models_c [A_q]B$.

We define that $G, x \not\models_c \forall R. A$ if $G, x \not\models_c [A_R]A$. \hfill \qed
Function \text{Find}(X)
1. if there exists \( z \in \Delta \setminus \Delta_0 \) with Label\((z) = X \) then return \( z \);
2. add a new element \( z \) to \( \Delta \) with \( \text{Label} (z) := X \);
3. return \( z \);

Procedure \text{ExtendLabel}(z, X)
1. if \( X \subseteq \text{Label}(z) \) then return;
2. if \( z \in \Delta_0 \) then \( \text{Label}(z) := \text{Satr}(\text{Label}(z) \cup X) \);
3. else
   4. \( z_* := \text{Find}(\text{Satr}(\text{Label}(z) \cup X)) \);
   5. \text{foreach} \( y, R, C \) such that \( \text{Next}(y, \exists R.C) = z \) do \( \text{Next}(y, \exists R.C) := z_* \);
   6. \text{foreach} \( y \) and \( R \) such that \( \text{LeastSucc}(y, R) = z \) do \( \text{LeastSucc}(y, R) := z_* \);

Function \text{CheckPremise}(x, C)
1. if \( C = \top \) then return \( true \);
2. else let \( C = C_1 \cap \ldots \cap C_k \);
3. \text{foreach} \( 1 \leq i \leq k \) do
   4. if \( C_i = A \) and \( A \notin \text{Label}(x) \) then return \( false \);
   5. else if \( C_i = \forall R.A \) and \( (\exists R.\top \notin \text{Label}(x) \lor \text{Next}(x, \exists R.\top) \text{ is not defined} \lor \text{A} \notin \text{Label}(\text{Next}(x, \exists R.\top))) \) then return \( false \);
   6. else if \( C_i = \exists R.A \) and \( \langle A_R \rangle A \notin \text{Label}(x) \) then return \( false \);
   7. else if \( C_i = \forall R.A \) and \( G, x \not\models_c \forall R.A \) then return \( false \);
8. return \( true \);

Algorithm 1: checking constructive satisfiability in Horn-CPDL\textsubscript{reg}
\textbf{Input:} a clausal Horn-CPDL\textsubscript{reg} knowledge base \( KB = (\mathcal{R}, \mathcal{T}, \mathcal{A}) \) and the RIA-automaton-specification \( \mathcal{A} \) of \( \mathcal{R} \).
\textbf{Output:} \text{true} if \( KB \) has a pseudo-model, or \text{false} otherwise.
\textbf{Data structure:} a Horn-CPDL\textsubscript{reg} graph \( G = (\Delta_0, \Delta, \text{Label}, \text{Next}, \text{LeastSucc}, \text{Status}) \).
1. let \( \Delta_0 \) be the set of all individuals occurring in \( \mathcal{A} \);
2. if \( \Delta_0 = \emptyset \) then \( \Delta_0 := \{\tau\} \);
3. \( \Delta := \Delta_0, \text{Status} := \text{unknown} \);
4. set \( \text{Next} \) and \( \text{LeastSucc} \) to the empty mappings;
5. \textbf{foreach} \( a \in \Delta_0 \) do
   6. \( \text{Label}(a) := \text{Satr}\{A \mid A(a) \in \mathcal{A} \cup \mathcal{T}\} \);
7. while some rule in Table 1 can make changes do
   8. choose such a rule and execute it; \hfill \text{// any strategy can be used}
   9. if \( \text{Status} = \text{unsat} \) then return \( false \);
10. return \( true \);
A pseudo-model extends \( z \) creates such a node \( z \) node \( z \) they should be merged. In other words, for every finite set \( \Delta \) nodes that are not named individuals occur in a tree or in different trees and have the same label, then \( \Delta \) may be infinite. However, we represent such a semi-forest as a finite graph with global caching: if two \( \Delta \) a Horn-CPDL \( x \) set \( \Delta \) of concepts. Consider the following cases:

- **Case** \( \forall \) if \( r(a, b) \in \mathcal{A} \) then
  
- **Algorithm 1** (on page 127) attempts to construct a pseudo-model of \( \mathcal{KB} \) by initializing a Horn-CPDL_\text{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 \( z \in \Delta \Delta_0 \) such that \( \text{Label}(z) = X \). The function \( \text{Find}(X) \) returns such a node \( z \) if it exists, or creates such a node \( z \) otherwise.

  For each \( x \in \Delta \), \( \text{Label}(x) \) 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** \( \forall \) if \( x \) is reachable from \( \Delta_0 \) and \( \text{Next}(x, \exists R.C) = y \) then
    
  - **Case** \( \exists \) if \( x \) is reachable from \( \Delta_0 \) and \( \text{LeastSucc}(x, R) = y \) then
    
  - **Case** \( \forall \) if \( x \) is reachable from \( \Delta_0 \) and \( \text{Next}(x, \exists R.C) = y \) then
    
  - **Case** \( \exists \) if \( x \) is reachable from \( \Delta_0 \) and \( \text{LeastSucc}(x, R) = y \) then
    
  - **Case** \( \exists \) if \( x \) is reachable from \( \Delta_0 \), \( \exists R.C \in \text{Label}(x), R \in \mathbb{R} \) and \( Next(x, \exists R.C) \) is not defined then
    
  - **Case** \( \exists \) if \( x \) is reachable from \( \Delta_0 \), \( R \in \mathbb{R} \) and \( \text{LeastSucc}(x, R) \) is not defined then
    
  - **Case** \( \exists \) if \( x \) is reachable from \( \Delta_0 \), \( C \subseteq D \in \text{Label}(x) \) and \( \text{CheckPremise}(x, C) \) then
    
  - **Case** \( \bot \) if \( \bot \in \text{Label}(x) \) or there exists \( \{A, \neg A\} \subseteq \text{Label}(x) \) then \( \text{Status} := \text{unsat} \);

\[
(\forall_1) \quad \text{if } r(a, b) \in \mathcal{A} \text{ then}
    
\[
(\forall_2) \quad \text{if } x \text{ is reachable from } \Delta_0 \text{ and } \text{Next}(x, \exists R.C) = y \text{ then}
    
\[
(\forall_3) \quad \text{if } x \text{ is reachable from } \Delta_0 \text{ and } \text{Next}(x, \exists R.C) = y \text{ then}
    
\[
(\forall_4) \quad \text{if } x \text{ is reachable from } \Delta_0 \text{ and } \text{LeastSucc}(x, R) = y \text{ then}
    
\[
(\forall_5) \quad \text{if } x \text{ is reachable from } \Delta_0 \text{ and } \text{LeastSucc}(x, R) = y \text{ then}
    
\[
(\exists) \quad \text{if } x \text{ is reachable from } \Delta_0, \exists R.C \in \text{Label}(x), R \in \mathbb{R} \text{ and } Next(x, \exists R.C) \text{ is not defined then}
    
\[
(\exists) \quad \text{if } x \text{ is reachable from } \Delta_0, R \in \mathbb{R} \text{ and } \text{LeastSucc}(x, R) \text{ is not defined then}
    
\[
(\exists) \quad \text{if } x \text{ is reachable from } \Delta_0, (C \subseteq D) \in \text{Label}(x) \text{ and } \text{CheckPremise}(x, C) \text{ then}
    
\[
(\bot) \quad \text{if } \bot \in \text{Label}(x) \text{ or there exists } \{A, \neg A\} \subseteq \text{Label}(x) \text{ then } \text{Status} := \text{unsat} ;
\]

Table 1. Expansion rules for Horn-CPDL_\text{reg} graphs.
Extending the label of $z$ for the above two cases is done by Procedure $\text{ExtendLabel}(z, X)$. 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 $\text{Next}(x, \exists R.C) = y$. Then, to realize the requirements at $x$, the label of $y$ should be extended with $X = \text{Trans}_2(\text{Label}(x), R)$. 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 $\text{Next}(x', \exists R'.C') = y$ 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 $(x, R, y)$ and reconnect $x$ to $y_* := \text{Find}(\text{Label}(y) \cup X)$ by setting $\text{Next}(x, \exists R.C) := y_*$. The extension is formally realized by the expansion rule $(\forall_2)$ (in Table 1). A similar case concerning $\text{LeastSucc}(x, R) = y$ is dealt with by the expansion rule $(\forall_4)$.

Consider the other expansion rules (in Table 1):

- $(\exists)$: If $\exists R.C \in \text{Label}(x)$ and $\text{Next}(x, \exists R.C)$ 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 $X = \text{Satr}(\{C\} \cup \text{Trans}_2(\text{Label}(x), R) \cup T)$ by setting $\text{Next}(x, \exists R.C) := \text{Find}(X)$.

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

- $(\exists)$: If $(C \sqsubseteq D) \in \text{Label}(x)$ and $C$ “holds” at $x$, then we extend the label of $x$ with $\{D\}$ by using the procedure $\text{ExtendLabel}$ discussed earlier. Suppose $C = C_1 \cap \ldots \cap C_k$. 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 $A \in \text{Label}(x)$.

  - Case $C_i = \forall R.A$ with $R \in R$: $C_i$ “holds” at $x$ if both $\forall R.A$ and $\exists R.T$ “hold” at $x$. If $\exists R.T$ “holds” at $x$ because the automaton $A_R$ 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 $(x_{j-1}, S_j, x_j)$ or $(x_j, s_j, x_{j-1})$ is a firm edge (belonging to $\text{Edges}$), then:

    * the automaton $A = A_R$ accepts $s_h \ldots s_1$ by a run $q_h, \ldots, q_0$;
    * since $\text{Label}(x_h)$ is saturated, $[A_R]_3 \exists R.T \in \text{Label}(x_h)$, which means $[A_{q_h}]_3 \exists R.T \in \text{Label}(x_h)$;
    * by the rules $(\forall_1)-(\forall_3)$, for each $j$ from $h - 1$ to 0, we can expect that $[A_{q_j}]_3 \exists R.T \in \text{Label}(x_j)$;
    * consequently, we can expect that $[A_{q_0}]_3 \exists R.T \in \text{Label}(x_0)$;
    * since $x = x_0$ and $q_0 \in F_A$, due to the saturation, we can expect that $\exists R.T \in \text{Label}(x)$ and $\text{Next}(x, \exists R.T)$ is defined.

Thus, to check whether $C_i$ “holds” at $x$ we just check whether $\exists R.T \in \text{Label}(x)$, $\text{Next}(x, \exists R.T)$ is defined and $A \in \text{Label}(\text{Next}(x, \exists R.T))$. The intuition is that $y =
Algorithm 1 runs in polynomial time in the size of the ABox $A$ 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. \hfill \Box

5.3. An Illustrative Example

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

\begin{align*}
A &\sqsubseteq \exists r.C \\
C &\sqsubseteq \forall \tau.D \\
D &\sqsubseteq C \\
A \cap \forall \exists r.C &\sqsubseteq E \\
A \cap \exists r.B &\sqsubseteq E \\
E &\sqsubseteq \forall (s \sqcup \tau)^*.F \\
\forall s^*.F &\sqsubseteq \bot.
\end{align*}

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

\begin{align*}
A_r &= (R, \{0, 1, 2\}, 0, \{(0, r, 1), (0, \tau, 2), (2, r, 2), (2, \tau, 2), (2, r, 1)\}, \{1\}) \\
A_\tau &= (R, \{0, 1, 2\}, 0, \{(0, \tau, 1), (0, r, 2), (2, r, 2), (2, \tau, 2), (2, r, 1)\}, \{1\}) \\
A_s &= (R, \{0, 1\}, 0, \{(0, s, 1)\}, \{1\}) \\
A_\bar{\pi} &= (R, \{0, 1\}, 0, \{(0, \bar{\pi}, 1)\}, \{1\}).
\end{align*}

We also have that:

\begin{align*}
A_s^* &= (\Sigma, \{0\}, 0, \{(0, s, 0)\}, \{0\}) \\
A_{(\bar{\pi}, s)}^* &= (\Sigma, \{0\}, 0, \{(0, s, 0), \{0, \bar{\pi}, 0\}\}, \{0\}).
\end{align*}

Consider the clausal Horn-CPDL$_{reg}$ knowledge base $KB = (\mathcal{R}, \mathcal{T}, A)$ with $A = \{A(a), B(a), A(b), r(a, b)\}$. 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 $\text{Next}(x, \exists r. \varphi) = y$ 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 $\text{LeastSucc}(x, s) = y$ and the edge was created at the step $n$. A label $n : \text{deleted}$ 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 (⊆) to the node \( x = b \) using the clause (15),
3: applying (∀₁) to the nodes \( a \) and \( b \) twice,
4: applying (∃) to the node \( x = a \) and the concept \( ∃ r.C \),
5: applying (∃) to the node \( x = b \) and the concept \( ∃ r.C \),
6: applying (⊆) to the node \( x = u \) using the clause (16),
7: applying (∀₂) to the nodes \( x = a \) and \( y = u' \),
8: applying (∀₂) to the nodes \( x = b \) and \( y = u' \),
9: applying (∃) to the node \( x = a \) and the concept \( ∃ r.⊤ \),
10: applying (⊆) to the node \( x = v \) using the clause (17),
11: applying (⊆) to the node \( x = a \) using the clause (18),
12: applying (⊆) 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 (⊆) to the node \( x = a \) using the clause (21).

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

\[ \Box \]

6. Concluding Remarks

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

\( \text{Horn-CPDL}_{reg} \) 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-T\_TEAMLOG [22], it is a useful rule language for formalizing agents’ cooperation.

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

In comparison with Horn-DL [25], apart from the concept constructor \( ∀∃R.C \) (for \( R \in \mathbb{R} \)), \( \text{Horn-CPDL}_{reg} \) also allows the concept constructor \( ∀R.C \) (for any role \( R \)) to appear on the LHS of TBox axioms. However, \( \text{Horn-CPDL}_{reg} \) is not more general than Horn-DL because the latter additionally allows nominals, qualified number restrictions, the \( ∃r.\text{Self} \) constructor, the universal role as well as assertions of the form \( \text{disjoint}(s, s') \), \( \text{irreflexive}(s) \), \( ¬s(a, b) \), \( a \neq b \). As future work, we will extend \( \text{Horn-CPDL}_{reg} \) with these features to obtain a rule language Horn-DL\_2 that is more general than Horn-DL, and hence also more general than Horn-SH\_\_IQ and Horn-SROIQ.

Our approach and method for \( \text{Horn-CPDL}_{reg} \) 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


Define \( \text{closure}_A(T) \) for a TBox \( T \) to be the smallest set of concepts such that:

- concepts and subconcepts occurring in \( T \) belong to \( \text{closure}_A(T) \),
- subconcepts occurring in \( \text{closure}_A(T) \) also belong to \( \text{closure}_A(T) \),
- \( \{[A_R]_3 R \top \mid R \in \mathcal{R}\} \subseteq \text{closure}_A(T) \),
- if \( \exists R.B \) occurs on the LHS of \( \sqsubseteq \) in a clause of \( T \), then \( [A_R]_3 \langle A_R \rangle B \in \text{Satr}(X) \),
- if \( \forall R.C \in \text{closure}_A(T) \), then \( [A_R]C \in \text{closure}_A(T) \),
- if \( [A]C \in \text{closure}_A(T) \) and \( q \in Q_A \), then \( [A_q]C \in \text{closure}_A(T) \),
- if \( [A]_3 C \in \text{closure}_A(T) \) and \( q \in Q_A \), then \( [A_q]_3 C \in \text{closure}_A(T) \).

Observe that \( \text{closure}_A(T) \) is finite.

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

**Proof:**

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

- Function \( \text{Find}(X) \) for \( X \subseteq \text{closure}_A(T) \) runs in constant time,
- Function \( \text{CheckPremise}(x, C) \) for a fixed \( C \) runs in \( O(n) \) steps,
• Procedure $\text{ExtendLabel}(z, X)$ runs in $O(n)$ steps for $X \subseteq \text{closure}_A(T)$,

• 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. $\text{Label}(a)$ for some $a \in \Delta_0$ is extended by a subset of $\text{closure}_A(T)$,

2. a new node $x$ is added to $\Delta$,

3. some $\text{Next}(x, \exists R.C)$ or $\text{LeastSucc}(x, R)$ is defined the first time,

4. some $\text{Next}(x, \exists R.C)$ or $\text{LeastSucc}(x, R)$ changes value from $y$ to some $y_* \in \Delta \setminus \Delta_0$ with $\text{Label}(y) \subset \text{Label}(y_*)$.

As the sizes of $\text{closure}_A(T)$, $\Delta \setminus \Delta_0$ and $\text{Label}(y)$ 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 $\Delta_0$, then unfolding the remaining part of the graph constructed by Algorithm 1, and then completing the interpretation of roles $R \in R$. For that we define $\Delta'$ and $\text{Edges}'$ as counterparts of $\Delta$ and $\text{Edges}$, respectively, together with a mapping $f : \Delta' \to \Delta$ and a queue $\text{unresolved}$ of elements of $\Delta'$ as follows:

• $\Delta' := \Delta_0$;

• $\text{FirmEdges} := \{ \langle a, r, b \rangle \mid r(a, b) \in A \}$, $\text{PseudoEdges} := \emptyset$;

• for each $a \in \Delta_0$, $f(a) := a$;

• add the elements of $\Delta_0$ to $\text{unresolved}$;

• while $\text{unresolved}$ is not empty:

  – extract an element $u$ from $\text{unresolved}$;

  – for each $\exists R.C \in \text{Label}(f(u))$:

    * add a new element $v$ to $\Delta'$ and $\text{unresolved}$;

    * $f(v) := \text{Next}(f(u), \exists R.C)$;

    * add $\langle u, R, v \rangle$ to $\text{FirmEdges}$;

  – for each $R \in R$ such that $\exists R. \top \notin \text{Label}(f(u))$:

    * add a new element $v$ to $\Delta'$ and $\text{unresolved}$;

    * $f(v) := \text{LeastSucc}(f(u), R)$;
The resulting data structures can be infinite. Let \( I \) be the pseudo-interpretation specified by:

- \( \Delta^I = \Delta' \);
- for each \( A \in C \), \( A^I = \{ u \in \Delta' \mid A \in \text{Label}(f(u)) \} \);
- for all \( R \in \mathbb{R} \), \( R^I \) are the least relations satisfying the following conditions:
  - \((R^I)^{-1} \subseteq R^I\),
  - if \( \langle u, R, v \rangle \in \text{FirmEdges} \), then \( \langle u, v \rangle \in R^I \),
  - for every word \( S_1 \ldots S_k \) accepted by \( A_R, S_1^I \circ \cdots \circ S_k^I \subseteq R^I \);
- for all \( R \in \mathbb{R} \), \( R^{I\varphi} \) are the least relations satisfying the following conditions:
  - \( R^I \subseteq R^{I\varphi} \) and \((R^{I\varphi})^{-1} \subseteq R^{I\varphi}\),
  - if \( \langle u, R, v \rangle \in \text{PseudoEdges} \), then \( \langle u, v \rangle \in R^{I\varphi} \),
  - for every word \( S_1 \ldots S_k \) accepted by \( A_R, S_1^{I\varphi} \circ \cdots \circ S_k^{I\varphi} \subseteq R^{I\varphi} \).

Observe that \( I \) is well-defined as it satisfies the condition that, for every \( x \in \Delta^I \) and every \( r \in \mathbb{R}_+ \), if \( Y = \{ y \mid \langle x, y \rangle \in r^I \} \neq \emptyset \), then \( \{ y \mid \langle x, y \rangle \in r^{I\varphi} \} = Y \). This is due to the use of \([A^I_0] \models \exists R. \top \) for the saturation operator, the expansion rules \((\forall_1) - (\forall_3)\) and the construction of the set \( \text{PseudoEdges} \).

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

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

Let \( A = A_R \). Since \( \varphi \in \text{Label}(f(u)) \) and \( \varphi = \forall R.A \), by saturation, we have that \([A^I_0]A \in \text{Label}(f(u)) \), which means \([A]A \in \text{Label}(f(u)) \) and \([A_{q_0}]A \in \text{Label}(f(u_0)) \). For each \( i \) from 1 to \( k \), due to the expansion rules \((\forall_1) - (\forall_3)\) and the saturation operator, it follows that \([A_{q_i}]A \in \text{Label}(f(u_i)) \). Since \( q_k \in F_A \) and \( u_k = v \), it follows that \( A \in \text{Label}(f(v)) \). Hence, \( v \in A^I \).
• Case \( \varphi = (C \subseteq D) \) and \( C = C_1 \cap \ldots \cap C_k \): Suppose \( u \in C^T \). We prove that \( u \in D^T \). The last call CheckPremise\((f(u), C)\) returned true because the following observations hold for every \( 1 \leq i \leq k \):

- Case \( C_i = A \): Since \( u \in C_i^T \), we have that \( A \in \text{Label}(f(u)) \).

- Case \( C_i = \exists R.A \): Since \( u \in C_i^T \), there exist a word \( S_1 \ldots S_h \) accepted by \( A_R \) and elements \( u_0 = u, u_1, \ldots, u_{h-1}, u_h \) such that \( u_h \in A^T \) and, for every \( 1 \leq j \leq h \):
  
  * \( u_{j-1} = u_j \) and \( S_j \) is of the form \((B?)\) with \( B \in \text{Label}(f(u_{j-1})) \);
  
  * \( (u_{j-1}, u_j) \in \text{FirmEdges} \) or \((u_j, \overline{S}_j, u_{j-1}) \in \text{FirmEdges} \).

The word \( \overline{S}_h \ldots \overline{S}_1 \) is accepted by \( A = A_R \) by a run \( q_h = q_A, q_{h-1}, \ldots, q_0 \). Since \( u_h \in A^T \), we have that \( A \in \text{Label}(f(u_h)) \) and, by saturation, \( [A_R]_{\overline{S}_j} \in \text{Label}(f(u_h)) \), which means \( [A_R]_{\overline{S}_j} \) \( A \in \text{Label}(f(u_h)) \). For each \( j \) from \( h \) to 1, due to the expansion rules \((\forall 1)-(\forall 3)\) and the saturation operator, it follows that \( [A_{u_{j-1}}]_{\overline{S}_j} \in \text{Label}(f(u_{j-1})) \).

- Case \( C_i = \forall \exists R.A \): With \( R \in R \), we have that \( u \in \langle 3R.T \rangle^T \). Thus, there exist a word \( S_1 \ldots S_h \) accepted by \( A_R \) and elements \( u_0 = u, u_1, \ldots, u_{h-1}, u_h \) such that, for every \( 1 \leq j \leq h \), \( (u_{j-1}, u_j) \in S_j^T \), and \((u_{j-1}, u_j) \in \text{FirmEdges} \) or \((u_j, \overline{S}_j, u_{j-1}) \in \text{FirmEdges} \).

The word \( \overline{S}_h \ldots \overline{S}_1 \) is accepted by \( A = A_R \) by a run \( q_h = q_A, q_{h-1}, \ldots, q_0 \). By saturation, \( [A_R]_{\overline{S}_j} \in \text{Label}(f(u_h)) \), which means \( [A_R]_{\overline{S}_j} \) \( A \in \text{Label}(f(u_h)) \). For each \( j \) from \( h \) to 1, due to the expansion rules \((\forall 1)-(\forall 3)\), it follows that \( [A_{u_{j-1}}]_{\overline{S}_j} \in \text{Label}(f(u_{j-1})) \). Since \( q_0 \in F_A \) and \( u_0 = u \), it follows that \( (A_R) \in \text{Label}(f(u)) \).

- Case \( C_i = \forall R.A \): Observe that, for any \( v \in \Delta' \), any finite automaton \( A \) over \( \Sigma \) and any \( B \in C \), if \((G, f(v)) \not\rightarrow_c A[B] \), then \( v \notin ([A]B)^T \). This can be proved by induction on the construction of the relation \( \not\rightarrow_c \). Since \( u \in C_i^T \), we have that \( u \in ([A]R)^T \), which implies \( G, f(u) \not\rightarrow_c A[R] \) and \( G, f(u) \not\rightarrow_c \forall R.A \).

We have shown that CheckPremise\((f(u), C)\) returned true. It follows that \( D \in \text{Label}(f(u)) \), and by the inductive assumption, \( u \in D^T \).

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

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 \( I \), every change made to the constructed graph can be justified by \( I \). 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\textsubscript{\text{reg}} knowledge base \(KB = \langle R, T, A \rangle\) has a pseudo-model \(I\). Consider an execution of Algorithm 1 for \(KB\) and any moment after executing the step 6 of that execution. Let \(Z = \{\langle x, u \rangle \in \Delta \times \Delta^I \mid u \in \text{Label}(x)\}\). Then:

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

Note that, if \(Z(x, u)\) holds, then \(u \in \text{Label}(x)\), which means \(\text{Label}(x)\) is satisfied at (and hence “justified by”) \(u\) in \(I\). 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 \(x \in \Delta \setminus \Delta_0\) was/is at some step reachable from \(\Delta_0\) and \(\text{Label}(x)\) was never changed.

**Corollary A.4.** If a clausal Horn-CPDL\textsubscript{\text{reg}} knowledge base \(KB\) has a pseudo-model, then Algorithm 1 returns \textit{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 \textit{false}. As it always terminates (by Lemma A.1), it must return \textit{true}. \(\square\)

**Lemma A.5.** Let \(KB = \langle R, T, A \rangle\) be a clausal Horn-CPDL\textsubscript{\text{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 \textit{true}. Let the sets \(\Delta', \text{FirmEdges}, \text{PseudoEdges} \) and the mapping \(f : \Delta' \to \Delta\) be constructed as in the proof of Lemma A.2. Let \(I\) be the interpretation specified by:

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

Similar to the proof of Lemma A.2, it can be proved that \(I\) is a model of \(KB\). \(\square\)