A Horn Fragment with PTime Data Complexity of Regular Description Logic with Inverse

Linh Anh Nguyen[a,b], Thi-Bich-Loc Nguyen[c], Andrzej Szałas[a,d]

[a]*Institute of Informatics, University of Warsaw*
[b]*Faculty of Information Technology, VNU University of Engineering and Technology*
[c]*Department of Information Technology, Hue University of Sciences*
[d]*Department of Computer and Information Science, Linköping University*

Abstract

We study a Horn fragment called Horn-$\mathcal{R}eg^I$ of the regular description logic with inverse $\mathcal{R}eg^I$, which extends the description logic $\mathcal{ALC}$ with inverse roles and regular role inclusion axioms characterized by finite automata. In contrast to the well-known Horn fragments $\mathcal{EL}$, DL-Lite, DLP, Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SROIQ}$ of description logics, Horn-$\mathcal{R}eg^I$ allows a form of the concept constructor "universal restriction" to appear at the left hand side of terminological inclusion axioms, while still has PTime data complexity. Namely, a universal restriction can be used in such places in conjunction with the corresponding existential restriction. We provide an algorithm with PTime data complexity for checking satisfiability of Horn-$\mathcal{R}eg^I$ knowledge bases.

Manuscript communication: received 16 December 2013, revised 27 April 2014, accepted 13 May 2014
Correspondence: Linh Anh Nguyen, nguyen@mimuw.edu.pl

Keywords: Description logics, Horn fragments, rule languages, Semantic Web.

1. Introduction

Description logics (DLs) are variants of modal logics suitable for expressing terminological knowledge. They represent the domain of interest in terms of individuals (objects), concepts and roles. A concept stands for a set of individuals, a role stands for a binary relation between individuals. The DL $\mathcal{SROIQ}$ [1] founds the logical base of the Web Ontology Language OWL 2, which was recommended by W3C as a layer for the architecture of the Semantic Web.

As reasoning in $\mathcal{SROIQ}$ has a very high complexity, W3C also recommended the profiles OWL 2 EL, OWL 2 QL and OWL 2 RL, which are based on the families of DLs $\mathcal{EL}$ [2, 3], DL-Lite [4, 5] and DLP [6]. These families of DLs are monotonic rule languages enjoying PTime data complexity. They are defined by selecting suitable Horn fragments of the corresponding full languages with appropriate restrictions adopted to eliminate nondeterminism. A number of Horn fragments of DLs with PTime data complexity have also been investigated in [7, 8, 9, 10, 11, 12, 13]. The combined complexities of Horn fragments of DLs were studied, amongst others, in [14]. Some Horn fragments of DLs without ABoxes that have PTime complexity have also been studied in [15, 2]. The fragments Horn-$\mathcal{SHIQ}$ [7, 11] and Horn-$\mathcal{SROIQ}$ [13] are notable, with considerable rich sets of allowed constructors and features. Combinations of rule languages like Datalog or its extensions with DLs have also been widely studied.

To eliminate nondeterminism, all $\mathcal{EL}$ [2, 3], DL-Lite [4, 5], DLP [6], Horn-$\mathcal{SHIQ}$ [7] and Horn-$\mathcal{SROIQ}$ [13] disallow (any form of) the universal restriction $\forall R.C$ at the left hand side of $\sqsubseteq$ in terminological axioms. The problem is that the general Horn fragment of the basic DL $\mathcal{ALC}$ allowing $\forall R.C$ at the left hand side of $\sqsubseteq$ has NP-complete data complexity [12]. Also, roles are not required to be serial (i.e., satisfying the condition $\forall x \exists y\, R(x, y)$), which complicates the construction of logically least models. For many application domains, the profiles OWL 2 EL, OWL 2 QL and OWL 2 RL languages and the underlying Horn fragments $\mathcal{EL}$, DL-Lite, DLP seem satisfactory. However, in general, forbidding $\forall R.C$ at the left hand side of $\sqsubseteq$ in terminological axioms is a serious restriction.

In [16] Nguyen introduced the deterministic Horn fragment of $\mathcal{ALC}$, where the constructor $\forall R.C$ is allowed at the left hand side of $\sqsubseteq$ in the combination with $\exists R.C$ (in the form $\forall R.C \sqcap \exists R.C$, denoted by $\forall\exists R.C$ [15]). He proved that such a fragment has PTime data complexity by providing a bottom-up method for constructing a logically least pseudo-model for a given deterministic positive knowledge base in the restricted language. In [12] Nguyen applied the method of [16] to regular DL $\mathcal{R}eg$, which extends $\mathcal{ALC}$ with regular role inclusion axioms characterized by finite automata. Let us denote the Horn fragment of $\mathcal{R}eg$ that allows the constructor $\forall\exists R.C$ at the left hand side of $\sqsubseteq$ by Horn-$\mathcal{R}eg$. As not every positive Horn-$\mathcal{R}eg$ knowledge base has a logically least model, Nguyen [12] proposed to approximate the instance checking problem in Horn-$\mathcal{R}eg$ by using its weakenings with PTime data complexity.

To see the usefulness of the constructor $\forall\exists R.C$ at the left hand side of $\sqsubseteq$ in terminological axioms, note that the following axioms are very intuitive and similar axioms are desirable:

$$\forall\exists hasChild.Happy \sqsubseteq HappyParent$$
$$\forall\exists hasChild.Male \sqsubseteq ParentWithOnlySons$$
$$\forall\exists hasChild.Female \sqsubseteq ParentWithOnlyDaughters$$
$$interesting \sqcap \forall\exists path.interesting \sqsubseteq perfect$$
$$interesting \sqcup \forall\exists link.interesting \sqsubseteq worth\_surfing.$$

The works [16, 12] found a starting point for the research concerning the universal restriction $\forall R.C$ at the left hand side of $\sqsubseteq$ in terminological axioms guaranteeing PTime data complexity. However, a big challenge is faced: the bottom-up approach is used, but not every positive Horn-$\mathcal{R}eg$ knowledge base has a logically least model. As a consequence, the work [12] on Horn-$\mathcal{R}eg$ is already complicated and the problem whether Horn-$\mathcal{R}eg$ has PTime data complexity remained open until [17].

This paper is a revised and extended version of our conference paper [17]. In this work we study a Horn fragment called Horn-$\mathcal{R}eg^I$ of the regular description logic with inverse $\mathcal{R}eg^I$. This fragment extends Horn-$\mathcal{R}eg$ with inverse roles. In contrast to the well-known Horn fragments $\mathcal{EL}$, DL-Lite, DLP, Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SROIQ}$ of description logics, Horn-$\mathcal{R}eg^I$ allows the concept constructor $\forall\exists R.C$ to appear at the left hand side of terminological inclusion axioms. We provide an algorithm with PTime data complexity for checking satisfiability of Horn-$\mathcal{R}eg^I$ knowledge bases. The key idea is to follow the top-down approach[1] and use a special technique to deal with non-seriality of roles.

The DL $\mathcal{R}eg^I$ (resp. $\mathcal{R}eg$) is a variant of regular grammar logic with (resp. without) converse [18, 19, 20, 21]. The current work is based on the previous works [16, 12, 22]. Namely, [22] considers Horn fragments of serial regular grammar logics with converse. The current work exploits the technique of [22] in dealing with converse (like inverse roles), but the difference is that it concerns *non-serial* regular DL with inverse roles. The change from grammar logic (i.e., modal logic) to DL is syntactic, but may increase the readability for the DL community.

The main achievements of the current paper are that:

- it overcomes the difficulties encountered in [16, 12] by using the top-down rather than bottom-up approach, and thus enables to show that both Horn-$\mathcal{R}eg$ and Horn-$\mathcal{R}eg^I$ have PTime data complexity, solving an open problem of [12];
- the technique introduced in the current paper for dealing with non-seriality leads to a solution for the important issue of allowing the concept constructor $\forall\exists R.C$ to appear at the left hand side of $\sqsubseteq$ in terminological inclusion axioms.

In comparison with [17], note that:

- Our algorithm now allows expansion rules to be applied in an arbitrary order. That is, any strategy can be used for expanding the constructed graph. This gives flexibility for optimizing the computation.
- The current paper provides full proofs for the results as well as additional examples and explanations.

The rest of this paper is structured as follows. In Section 2 we present notation and semantics of $\mathcal{R}eg^I$ and recall automaton-modal operators. In Section 3 we define the Horn-$\mathcal{R}eg^I$ fragment. In Section 4 we present our algorithm of checking satisfiability of Horn-$\mathcal{R}eg^I$ knowledge bases and discuss our technique of dealing with $\forall\exists R.C$ at the left hand side of $\sqsubseteq$. In Section 5 we give proofs for the properties of the algorithm. We conclude this work in Section 6.

[1]In the top-down approach, the considered query is negated and added into the knowledge base, and in general, a knowledge base may contain "negative" constraints.

2. Preliminaries

2.1. Notation and Semantics of $\mathcal{Reg}^I$

Our language uses a countable set $\mathbf{C}$ of *concept names*, a countable set $\mathbf{R}_+$ of *role names*, and a countable 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.

For $r \in \mathbf{R}_+$, we call the expression $\overline{r}$ the *inverse* of r. Let $\mathbf{R}_- = \{\overline{r} \mid r \in \mathbf{R}_+\}$ and $\mathbf{R} = \mathbf{R}_+ \cup \mathbf{R}_-$. For $R = \overline{r}$, let $\overline{R}$ stand for r. We call elements of $\mathbf{R}$ *roles* and use letters like R, S to denote them.

A *context-free semi-Thue system* $\mathcal{S}$ over $\mathbf{R}$ is a finite set of context-free production rules over alphabet $\mathbf{R}$. 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 $\mathcal{S}$.[2] 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 $k \geq 0$. In the case $k = 0$, the left hand side 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.

Example 1. Let $\mathcal{R} = \{\overline{r} \circ r \sqsubseteq r,\ \overline{r} \circ r \sqsubseteq \overline{r}\}$. The symmetric regular semi-Thue system corresponding to $\mathcal{R}$ is

$$\mathcal{S} = \{r \to \overline{r}r,\ \ \overline{r} \to \overline{r}r\}.$$

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

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:

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

Observe that every regular set of RIAs in $\mathcal{SROIQ}$ [1] and Horn-$\mathcal{SROIQ}$ [13] 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 $\mathcal{SROIQ}$ [1] and Horn-$\mathcal{SROIQ}$ [13] allows only a bounded nesting depth of the star operator *, while "regularity" of a regular RBox in Horn-$\mathcal{Reg}^I$ is not so restricted. That is, our notion of regular RBox is more general than the notion of regular set of RIAs in $\mathcal{SROIQ}$ [1] and Horn-$\mathcal{SROIQ}$ [13]. $\triangleleft$

Let $\mathcal{R}$ be a regular RBox and $\mathbf{A}$ be its RIA-automaton-specification. For $R, S \in \mathbf{R}$, we say that R is a *subrole* of S w.r.t. $\mathcal{R}$, denoted by $R \sqsubseteq_{\mathcal{R}} S$, if the word R is accepted by $\mathbf{A}_S$.

Concepts are defined by the following BNF grammar, where $A \in \mathbf{C}$, $R \in \mathbf{R}$:

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C$$

We use letters like C, D to denote concepts (including complex concepts).

A *TBox* is a finite set of *TBox axioms* of the form $C \sqsubseteq D$. An *ABox* is a finite set of *assertions* of the form $C(a)$ or $r(a,b)$. A *knowledge base* is a tuple $\langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$, where $\mathcal{R}$ is a regular RBox, $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox.

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 $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.

Define

$$\begin{aligned}
(\overline{r})^\mathcal{I} &= (r^\mathcal{I})^{-1} = \{\langle y,x\rangle \mid \langle x,y\rangle \in r^\mathcal{I}\} \text{ (for } r \in \mathbf{R}_+) \\
\varepsilon^\mathcal{I} &= \{\langle x,x\rangle \mid x \in \Delta^\mathcal{I}\}.
\end{aligned}$$

The interpretation function $\cdot^\mathcal{I}$ is extended to complex concepts as follows:

$$\top^\mathcal{I} = \Delta^\mathcal{I}, \quad \bot^\mathcal{I} = \emptyset, \quad (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I},$$
$$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}, \quad (C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I},$$
$$(\forall R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \forall y\,(\langle x, y\rangle \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I})\},$$
$$(\exists R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y\,(\langle x, y\rangle \in R^\mathcal{I} \land y \in C^\mathcal{I})\}.$$

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

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

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

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 $\langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ if it is a model of all $\mathcal{R}$, $\mathcal{T}$ 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.

2.2. Automaton-Modal Operators

Given an interpretation $\mathcal{I}$ and a finite automaton A over alphabet $\mathbf{R}$, define $\mathsf{A}^\mathcal{I} = \{\langle x, y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid$ 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 $\langle x_{i-1}, x_i\rangle \in R_i^\mathcal{I}$ for all $1 \le i \le k\}$.

We will use auxiliary modal operators $[\mathsf{A}]$ and $\langle \mathsf{A}\rangle$, where A is a finite automaton over alphabet $\mathbf{R}$. We call $[\mathsf{A}]$ (resp. $\langle \mathsf{A}\rangle$) a *universal* (resp. *existential*) *automaton-modal operator*. Automaton-modal operators were used earlier, among others, in [23, 20, 24, 25, 12].

In the *extended language*, if C is a concept then $[\mathsf{A}]C$ and $\langle \mathsf{A}\rangle C$ are also concepts. The semantics of $[\mathsf{A}]C$ and $\langle \mathsf{A}\rangle C$ are defined as follows:

$$([\mathsf{A}]C)^\mathcal{I} = \Big\{x \in \Delta^\mathcal{I} \mid \forall y\Big(\langle x, y\rangle \in \mathsf{A}^\mathcal{I} \text{ implies } y \in C^\mathcal{I}\Big)\Big\}$$
$$(\langle \mathsf{A}\rangle C)^\mathcal{I} = \Big\{x \in \Delta^\mathcal{I} \mid \exists y\Big(\langle x, y\rangle \in \mathsf{A}^\mathcal{I} \text{ and } y \in C^\mathcal{I}\Big)\Big\}.$$

For a finite automaton A over $\mathbf{R}$, let the components of A be denoted as in the following:

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

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

1. $(\forall R.C)^\mathcal{I} = ([\mathbf{A}_R]C)^\mathcal{I}$,
2. $(\exists R.C)^\mathcal{I} = (\langle \mathbf{A}_R\rangle C)^\mathcal{I}$,
3. $C^\mathcal{I} \subseteq ([\mathbf{A}_{\overline{R}}]\langle \mathbf{A}_R\rangle C)^\mathcal{I}$,
4. $C^\mathcal{I} \subseteq ([\mathbf{A}_{\overline{R}}]\exists R.C)^\mathcal{I}$.

Proof: The first assertion holds because the following conditions are equivalent:

- $x \in (\forall R.C)^\mathcal{I}$;
- for all $y \in \Delta^\mathcal{I}$, if $\langle x, y\rangle \in R^\mathcal{I}$ then $y \in C^\mathcal{I}$;
- for all $y \in \Delta^\mathcal{I}$, if $\langle x, y\rangle \in (\mathbf{A}_R)^\mathcal{I}$ then $y \in C^\mathcal{I}$;
- $x \in ([\mathbf{A}_R]C)^\mathcal{I}$.

Analogously, the second assertion holds.

Consider the third assertion and suppose $x \in C^\mathcal{I}$. We show that $x \in ([\mathbf{A}_{\overline{R}}]\langle \mathbf{A}_R\rangle C)^\mathcal{I}$. Let y be an arbitrary element of $\Delta^\mathcal{I}$ such that $\langle x, y\rangle \in (\mathbf{A}_{\overline{R}})^\mathcal{I}$. By definition, there exist a word $R_1 \ldots R_k$ accepted by $\mathbf{A}_{\overline{R}}$ and elements $x_0 = x, x_1, \ldots, x_k = y$ of $\Delta^\mathcal{I}$ such that $\langle x_{i-1}, x_i\rangle \in R_i^\mathcal{I}$ for all $1 \le i \le k$. Observe that the word $\overline{R}_k \ldots \overline{R}_1$ is accepted by $\mathbf{A}_R$. Since $x \in C^\mathcal{I}$, $x_k = y$, $x_0 = x$ and $\langle x_i, x_{i-1}\rangle \in \overline{R}_i^\mathcal{I}$ for all $k \ge i \ge 1$, we have that $y \in (\langle \mathbf{A}_R\rangle C)^\mathcal{I}$. Therefore, $x \in ([\mathbf{A}_{\overline{R}}]\langle \mathbf{A}_R\rangle C)^\mathcal{I}$.

The fourth assertion directly follows from the third and second assertions. $\triangleleft$

3. The Horn-$\mathcal{R}eg^I$ Fragment

Let $\forall\exists R.C$ stand for $\forall R.C \sqcap \exists R.C$. *Left-hand-side Horn-$\mathcal{R}eg^I$ concepts*, called *LHS Horn-$\mathcal{R}eg^I$ concepts* for short, are defined by the following grammar, where $A \in \mathbf{C}$ and $R \in \mathbf{R}$:

$$C ::= \top \mid A \mid C \sqcap C \mid C \sqcup C \mid \forall\exists R.C \mid \exists R.C$$

Right-hand-side Horn-$\mathcal{R}eg^I$ concepts, called *RHS Horn-$\mathcal{R}eg^I$ concepts* for short, are defined by the following BNF grammar, where $A \in \mathbf{C}$, D is an LHS Horn-$\mathcal{R}eg^I$ concept, and $R \in \mathbf{R}$:

$$C ::= \top \mid \bot \mid A \mid \neg D \mid C \sqcap C \mid \neg D \sqcup C \mid \forall R.C \mid \exists R.C$$

A *Horn-$\mathcal{R}eg^I$ TBox axiom*, is an expression of the form $C \sqsubseteq D$, where C is an LHS Horn-$\mathcal{R}eg^I$ concept and D is an RHS Horn-$\mathcal{R}eg^I$ concept.

A *Horn-$\mathcal{R}eg^I$ TBox* is a finite set of *Horn-$\mathcal{R}eg^I$ TBox axioms*.

A *Horn-$\mathcal{R}eg^I$ clause* is a Horn-$\mathcal{R}eg^I$ 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, $\forall\exists R.A$ or $\exists R.A$,
- D is of the form $\bot$, A, $\forall R.A$ or $\exists R.A$,
- $k \geq 1$, $A \in \mathbf{C}$ and $R \in \mathbf{R}$.

A *clausal Horn-$\mathcal{R}eg^I$ TBox* is a TBox consisting of Horn-$\mathcal{R}eg^I$ clauses.

A *Horn-$\mathcal{R}eg^I$ ABox* is a finite set of assertions of the form $C(a)$ or $r(a,b)$, where C is an RHS Horn-$\mathcal{R}eg^I$ concept. A *reduced ABox* is a finite set of assertions of the form $A(a)$ or $r(a,b)$.

A knowledge base $\langle\mathcal{R},\mathcal{T},\mathcal{A}\rangle$ is called a *Horn-$\mathcal{R}eg^I$ knowledge base* if $\mathcal{T}$ is a Horn-$\mathcal{R}eg^I$ TBox and $\mathcal{A}$ is a Horn-$\mathcal{R}eg^I$ ABox. When $\mathcal{T}$ is a clausal Horn-$\mathcal{R}eg^I$ TBox and $\mathcal{A}$ is a reduced ABox, we call such a knowledge base a *clausal Horn-$\mathcal{R}eg^I$ knowledge base*.

Example 2. This example is about Web pages. Let $\mathbf{R}_+ = \{link, path\}$ and let $\mathcal{R}$ be the regular RBox consisting of the following role axioms:

$$\begin{array}{ll} link \sqsubseteq path, & \overline{link} \sqsubseteq \overline{path}, \\ link \circ path \sqsubseteq path, & \overline{path} \circ \overline{link} \sqsubseteq \overline{path}. \end{array}$$

This RBox "defines" *path* to be the transitive closure of *link*. As the RIA-automaton-specification of $\mathcal{R}$ we can take the mapping $\mathbf{A}$ such that:

$$\begin{array}{rcl} \mathbf{A}_{link} & = & \langle \mathbf{R}, \{1,2\}, 1, \{\langle 1, link, 2\rangle\}, \{2\}\rangle, \\ \mathbf{A}_{\overline{link}} & = & \langle \mathbf{R}, \{1,2\}, 2, \{\langle 2, \overline{link}, 1\rangle\}, \{1\}\rangle, \\ \mathbf{A}_{path} & = & \langle \mathbf{R}, \{1,2\}, 1, \\ & & \{\langle 1, link, 1\rangle, \langle 1, link, 2\rangle, \langle 1, path, 2\rangle\}, \{2\}\rangle, \\ \mathbf{A}_{\overline{path}} & = & \langle \mathbf{R}, \{1,2\}, 2, \\ & & \{\langle 1, \overline{link}, 1\rangle, \langle 2, \overline{link}, 1\rangle, \langle 2, \overline{path}, 1\rangle\}, \{1\}\rangle. \end{array}$$

Let $\mathcal{T}$ be the TBox consisting of the following program clauses:

$$\begin{array}{l} perfect \sqsubseteq interesting \sqcap \forall path.interesting \\ interesting \sqcap \forall\exists path.interesting \sqsubseteq perfect \\ interesting \sqcup \forall\exists link.interesting \sqsubseteq worth\_surfing. \end{array}$$

Let $\mathcal{A}$ be the ABox specified by the concept assertion *perfect*(b) and the following role assertions of *link*:

$$\begin{tikzcd} & & a \arrow[dl] \arrow[dd] & & b \arrow[dl] \arrow[dr] & & \\ & c \arrow[dl] & & e \arrow[dl] & & f \arrow[dr] & \\ g & & h & & & & i \arrow[uull, bend right] \end{tikzcd}$$

Then $KB = \langle\mathcal{R},\mathcal{T},\mathcal{A}\rangle$ is a Horn-$\mathcal{R}eg^I$ knowledge base. (Ignoring $\overline{link}$ and $\overline{path}$, which are not essential in this example, KB can be treated as a Horn-$\mathcal{R}eg$ knowledge base.) It can be seen that b, e, f, i are instances of the concepts *perfect*, *interesting*, *worth_surfing* w.r.t. KB. Furthermore, h is also an instance of the concept *interesting* w.r.t. KB. $\lhd$

The *length* of a concept, an assertion or an axiom φ is the number of symbols occurring in φ. The *size* of an ABox is the sum of the lengths of its assertions. The *size* of a TBox is the sum of the lengths of its axioms.

The *data complexity* class of Horn-$\mathcal{R}eg^I$ is defined to be the complexity class of the problem of checking satisfiability of a Horn-$\mathcal{R}eg^I$ knowledge base $\langle\mathcal{R},\mathcal{T},\mathcal{A}\rangle$, measured in the size of $\mathcal{A}$ when assuming that $\mathcal{R}$ and $\mathcal{T}$ are fixed and $\mathcal{A}$ is a reduced ABox.

Proposition 2. Let $KB = \langle\mathcal{R},\mathcal{T},\mathcal{A}\rangle$ be a Horn-$\mathcal{R}eg^I$ knowledge base.

1. If C is an LHS Horn-$\mathcal{R}eg^I$ concept then $KB \models C(a)$ iff the Horn-$\mathcal{R}eg^I$ knowledge base $\langle\mathcal{R},\mathcal{T}\cup\{C \sqsubseteq A\}, \mathcal{A}\cup\{\neg A(a)\}\rangle$ is unsatisfiable, where A is a fresh concept name.
2. KB can be converted in polynomial time in the sizes of $\mathcal{T}$ and $\mathcal{A}$ to a Horn-$\mathcal{R}eg^I$ knowledge base $KB' = \langle\mathcal{R},\mathcal{T}',\mathcal{A}'\rangle$ with $\mathcal{A}'$ being a reduced ABox such that KB is satisfiable iff KB' is satisfiable.
3. KB can be converted in polynomial time in the size of $\mathcal{T}$ to a Horn-$\mathcal{R}eg^I$ knowledge base $KB' = \langle\mathcal{R},\mathcal{T}',\mathcal{A}\rangle$ with $\mathcal{T}'$ being a clausal Horn-$\mathcal{R}eg^I$ TBox such that KB is satisfiable iff KB' is satisfiable.

Proof: The first assertion is clear. For the second assertion, we start with $\mathcal{T}' := \mathcal{T}$ and $\mathcal{A}' := \mathcal{A}$ and then modify them as follows: for each $C(a) \in \mathcal{A}'$ where C is not a concept name, replace $C(a)$ in $\mathcal{A}'$ by $A(a)$, where A is a fresh concept name, and add to $\mathcal{T}'$ the axiom $A \sqsubseteq C$. It is easy to check that the resulting Horn-$\mathcal{R}eg^I$ knowledge base $KB' = \langle\mathcal{R},\mathcal{T}',\mathcal{A}'\rangle$ is satisfiable iff KB is satisfiable.

For the third assertion, we apply the technique that replaces complex concepts by fresh concept names. For example, if $\forall\exists R.C \sqsubseteq \exists S.D$ is an axiom of $\mathcal{T}$, where C and D are complex concepts, then we replace it by axioms $C \sqsubseteq A_C$, $\forall\exists R.A_C \sqsubseteq \exists S.A_D$ and $A_D \sqsubseteq D$, where A_C and A_D are fresh concept names. $\lhd$

Corollary 3. Every Horn-$\mathcal{R}eg^I$ knowledge base $KB = \langle\mathcal{R},\mathcal{T},\mathcal{A}\rangle$ can be converted in polynomial time in the sizes of $\mathcal{T}$ and $\mathcal{A}$ to a clausal Horn-$\mathcal{R}eg^I$ knowledge base $KB' = \langle \mathcal{R}, \mathcal{T}', \mathcal{A}' \rangle$ such that KB is satisfiable iff KB' is satisfiable.

Proof: This corollary follows from the second and third assertions of Proposition 2. In particular, we first apply the conversion mentioned in the second assertion of Proposition 2 to KB to obtain KB_2, and then apply the conversion mentioned in the third assertion of Proposition 2 to KB_2 to obtain KB'. $\triangleleft$

4. Checking Satisfiability of Horn-$\mathcal{R}eg^I$ Knowledge Bases

In this section we present an algorithm that, given a clausal Horn-$\mathcal{R}eg^I$ knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ together with the RIA-automaton-specification $\mathbf{A}$ of $\mathcal{R}$, checks whether the knowledge base is satisfiable. The algorithm has PTime data complexity.

We will treat each TBox axiom $C \sqsubseteq D$ from $\mathcal{T}$ as a concept standing for a global assumption. That is, $C \sqsubseteq D$ is logically equivalent to $\neg C \sqcup D$, and it is a global assumption for an interpretation $\mathcal{I}$ if $(\neg C \sqcup D)^{\mathcal{I}} = \Delta^{\mathcal{I}}$.

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. if $\forall R.C \in \mathsf{Satr}(X)$ then $[\mathbf{A}_R]C \in \mathsf{Satr}(X)$,
2. if $[\mathsf{A}]C \in \mathsf{Satr}(X)$ and $q_{\mathsf{A}} \in F_{\mathsf{A}}$ then $C \in \mathsf{Satr}(X)$,
3. if $\forall\exists R.A$ occurs in $\mathcal{T}$ for some A then $[\mathbf{A}_{\overline{R}}]\exists R.\top \in \mathsf{Satr}(X)$,
4. if $A \in \mathsf{Satr}(X)$ and $\exists R.A$ occurs at the left hand side of $\sqsubseteq$ in some clause of $\mathcal{T}$ then $[\mathbf{A}_{\overline{R}}]\langle \mathbf{A}_R \rangle A \in \mathsf{Satr}(X)$.

Notice the third item in the above list. It is used for dealing with non-seriality and the concept constructor $\forall\exists R.A$. Another treatment for the problem of non-seriality and $\forall\exists R.A$ is the step 5 of Function `CheckPremise` (used in our algorithm). It will be explained later.

For $R \in \mathbf{R}$, the *transfer* of X through R is

$$\mathsf{Trans}(X, R) = \{[\mathsf{A}_q]C \mid [\mathsf{A}]C \in X \text{ and } \langle q_{\mathsf{A}}, R, q \rangle \in \delta_{\mathsf{A}}\}.$$

Our algorithm for checking satisfiability of $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ uses the data structure $\langle \Delta_0, \Delta, Label, Next \rangle$, which is called a *Horn-$\mathcal{R}eg^I$ graph*, where:

- Δ_0 : the set of all individual names occurring in $\mathcal{A}$,
- Δ : a set of objects including Δ_0,
- *Label* : a function mapping each $x \in \Delta$ to a set of concepts,

$(\forall_i)$	**if** $r(a,b) \in \mathcal{A}$ **then** `ExtendLabel`$(b, \mathsf{Trans}(Label(a), r))$;
$(\forall)$	**if** x is reachable from Δ_0 and $Next(x, \exists R.C) = y$ **then** $Next(x, \exists R.C) :=$ `Find`$(Label(y) \cup \mathsf{Satr}(\mathsf{Trans}(Label(x), R)))$;
$(\forall_I)$	**if** x is reachable from Δ_0 and $\langle x, R, y \rangle \in Edges$ **then** `ExtendLabel`$(x, \mathsf{Trans}(Label(y), \overline{R}))$;
$(\exists)$	**if** x is reachable from Δ_0, $\exists R.C \in Label(x)$, $R \in \mathbf{R}$ and $Next(x, \exists R.C)$ is not defined **then** $Next(x, \exists R.C) :=$ `Find`$(\mathsf{Satr}(\{C\} \cup \mathsf{Trans}(Label(x), R)) \cup \mathcal{T}')$;
$(\sqsubseteq)$	**if** x is reachable from Δ_0, $(C \sqsubseteq D) \in Label(x)$ and `CheckPremise`(x, C) **then** `ExtendLabel`$(x, \{D\})$;

Table 1: Expansion rules for Horn-$\mathcal{R}eg^I$ graphs.

Function Find(X)

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

Procedure ExtendLabel(z, X)

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

Function CheckPremise(x, C)

```
1 if $C = \top$ then return true
2 else let $C = C_1 \sqcap \ldots \sqcap C_k$;
3 foreach $1 \leq i \leq k$ do
4     if $C_i = A$ and $A \notin Label(x)$ then return false
5     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
6         return false
7     else if $C_i = \exists R.A$ and $\langle \mathbf{A}_R \rangle A \notin Label(x)$ then
8         return false
9 return true
```

Algorithm 1: checking satisfiability in Horn-$\mathcal{R}eg^I$

```
Input: a clausal Horn-$\mathcal{R}eg^I$ knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$
       and the RIA-automaton-specification $\mathbf{A}$ of $\mathcal{R}$.
Output: true if $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ is satisfiable,
        or false otherwise.
let $\Delta_0$ be the set of all individuals occurring in $\mathcal{A}$;
if $\Delta_0 = \emptyset$ then $\Delta_0 := \{\tau\}$;
$\Delta := \Delta_0$, $\mathcal{T}' := \mathsf{Satr}(\mathcal{T})$, empty the mapping $Next$;
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 there exists $x \in \Delta$ such that $\bot \in Label(x)$ then
        return false
return true
```

- $Next : \Delta \times \{\exists R.\top, \exists R.A \mid R \in \mathbf{R}, A \in \mathbf{C}\} \to \Delta$ is a partial mapping.

For $x \in \Delta$, $Label(x)$ is called the *label* of x. A fact $Next(x, \exists R.C) = y$ means that $\exists R.C \in Label(x)$, $C \in Label(y)$, and $\exists R.C$ is "realized" at x by going to y. When defined, $Next(x, \exists R.\top)$ denotes the "logically smallest R-successor of x".

Define

$$Edges = \{\langle x, R, y\rangle \mid R(x, y) \in \mathcal{A} \text{ or } Next(x, \exists R.C) = y \text{ for some } C\}.$$

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 $\langle x_{i-1}, R_i, x_i\rangle \in Edges$ for all $1 \leq i \leq k$.

Algorithm 1 attempts to construct a model of $\langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ by initializing a Horn-$\mathcal{R}eg^I$ graph and then expanding it by the rules in Table 1. The intended 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 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 $Label(z) = X$. The function `Find`(X) returns such a node z if it exists, or creates such a node z otherwise. A tuple $\langle x, R, y\rangle \in Edges$ represents an edge $\langle x, y\rangle$ with label R of the graph. The notions of *predecessor* and *successor* are defined as usual.

For each $x \in \Delta$, $Label(x)$ 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 $Label(z)$ 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 $z_* :=$ `Find`$(Label(z) \cup \mathsf{Satr}(X))$ for playing the role of z. In particular, for each y, R and C such that $Next(y, \exists R.C) = z$, we set $Next(y, \exists R.C) := z_*$.

Extending the label of z for the above two cases is done by Procedure `ExtendLabel`(z, X). The third case is considered below.

Suppose that $Next(x, \exists R.C) = y$. Then, to realize the requirements at x, the label of y should be extended with $X = \mathsf{Satr}(\mathsf{Trans}(Label(x), R))$. How can we realize such an extension? Recall that we intend to construct a forest-like model for $\langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$, but use global caching to guarantee termination. There may exist another $Next(x', \exists R'.C') = y$ 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 $y_* :=$ `Find`$(Label(y) \cup X)$ by setting $Next(x, \exists R.C) := y_*$. The extension is formally realized by the expansion rule $(\forall)$ (in Table 1).

Consider the other expansion rules (in Table 1):

- $(\forall_i)$: If $r(a, b) \in \mathcal{A}$ then we extend $Label(b)$ with $\mathsf{Satr}(\mathsf{Trans}(Label(a), R))$.

- $(\forall_I)$: If $\langle x, R, y\rangle \in Edges$ then we extend the label of x with $\mathsf{Trans}(Label(y), \overline{R})$ by using the procedure `ExtendLabel` discussed earlier.

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

- $(\sqsubseteq)$: If $(C \sqsubseteq D) \in Label(x)$ 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 $A \in Label(x)$.
  - Case $C_i = \forall\exists R.A$: C_i "holds" at x if both $\forall R.A$ and $\exists R.\top$ "hold" at x. If $\exists R.\top$ "holds" at x by the evidence of a path connecting x to a node z with (forward or backward) "edges" labeled by $S_1, \ldots, S_k$ such that the word $S_1 \ldots S_k$ is accepted by the automaton $\mathsf{A} = \mathbf{A}_R$, that is:
    - ∗ there exist nodes $x_0, \ldots, x_k$ such that $x_0 = x$, $x_k = z$ and, for $1 \leq j \leq k$, either $\langle x_{j-1}, S_j, x_j\rangle \in Edges$ or $\langle x_j, \overline{S}_j, x_{j-1}\rangle \in Edges$,

- $*$ there exist states $q_0, \ldots, q_k$ of A such that $q_0 = q_{\mathsf{A}}$, $q_k \in Q_{\mathsf{A}}$ and, for $1 \le j \le k$, $\langle q_{j-1}, S_j, q_j \rangle \in \delta_{\mathsf{A}}$,

then, with $\overline{\mathsf{A}} = \mathbf{A}_{\overline{R}}$, we have that:

- $*$ since $Label(z)$ is saturated, $[\mathbf{A}_{\overline{R}}]\exists R.\top \in Label(z)$, i.e. $[\overline{\mathsf{A}}_{q_k}]\exists R.\top \in Label(x_k)$,
- $*$ by the rules $(\forall_i)$, $(\forall)$ and $(\forall_I)$ (listed in Table 1 and used in Algorithm 1), for each j from $k-1$ to 0, we can expect that $[\overline{\mathsf{A}}_{q_j}]\exists R.\top \in Label(x_j)$,
- $*$ consequently, since $q_0 = q_{\mathsf{A}} \in Q_{\overline{\mathsf{A}}}$, due to the saturation we can expect that $\exists R.\top \in Label(x_0)$.

That is, we can expect that $\exists R.\top \in Label(x)$ and $Next(x, \exists R.\top)$ is defined. To check whether C_i "holds" at x we just check whether $\exists R.\top \in Label(x)$, $Next(x, \exists R.\top)$ is defined and $A \in Label(Next(x, \exists R.\top))$. The intuition is that, $y = Next(x, \exists R.\top)$ is the "least R-successor" of x, and if $A \in Label(y)$ then A will occur in all R-successors of x.

– Case $C_i = \exists R.A$: If $\exists R.A$ "holds" at x by the evidence of a path connecting x to a node z with (forward or backward) "edges" labeled by $S_1, \ldots, S_k$ such that the word $S_1 \ldots S_k$ is accepted by $\mathbf{A}_R$ and $A \in Label(z)$ then, since $[\mathbf{A}_{\overline{R}}]\langle \mathbf{A}_R \rangle A$ is included in $Label(z)$ by saturation, we can expect that $\langle \mathbf{A}_R \rangle A \in Label(x)$. 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 $\exists R.A$.) The reason for using this technique is due to the use of global caching (in order to guarantee termination).

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 rule $(\forall)$ (specified in Table 1). Formally, checking whether the premise C of a Horn-$\mathcal{R}eg^I$ clause $C \sqsubseteq D$ "holds" at x is done by Function `CheckPremise`(x, C).

Expansions by modifying the label of a node and/or setting the mapping $Next$ 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 some $x \in \Delta$ has $Label(x)$ containing $\bot$, Algorithm 1 returns *false*, which means that the knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable. When the graph cannot be expanded any more, the algorithm terminates in the normal mode with result *true*, which means $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ is satisfiable.

Theorem 4. Algorithm 1 correctly checks satisfiability of clausal Horn-$\mathcal{R}eg^I$ knowledge bases and has PTIME data complexity.

This theorem follows from Lemmas 6, 7 and Corollary 9, which are given and proved in the next section. The following corollary follows from this theorem and Proposition 2.

Corollary 5. The problem of checking satisfiability of Horn-$\mathcal{R}eg^I$ knowledge bases has PTIME data complexity.

Example 3. Let $\mathbf{R}_+ = \{r\}$, $\mathbf{C} = \{A, B, C, D, E\}$, $\mathbf{I} = \{a, b\}$, $\mathcal{R} = \{\overline{r} \circ r \sqsubseteq r,\ \overline{r} \circ r \sqsubseteq \overline{r}\}$, and let $\mathcal{T}$ be the TBox consisting of the following axioms:

$$\begin{aligned} A &\sqsubseteq \exists r.C && (1)\\ C &\sqsubseteq \forall \overline{r}.D && (2)\\ D &\sqsubseteq C && (3)\\ A \sqcap \forall\exists r.C &\sqsubseteq E && (4)\\ A \sqcap \exists r.B &\sqsubseteq E && (5)\\ E &\sqsubseteq \bot. && (6) \end{aligned}$$

As discussed in Example 1, $\mathcal{R}$ is a regular RBox with the following RIA-automaton-specification:

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

Note that $\mathbf{A}_r = (\mathbf{A}_r)_0$ and $\mathbf{A}_{\overline{r}} = (\mathbf{A}_{\overline{r}})_0$.

Consider the Horn-$\mathcal{R}eg^I$ knowledge base $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ with $\mathcal{A} = \{A(a),\ B(a),\ A(b),\ r(a,b)\}$.

Figure 1 illustrates the Horn-$\mathcal{R}eg^I$ graph constructed by Algorithm 1 for KB. The nodes of the graph are a, b, u, u', v, v', 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 13. In the table representing a node $x \in \{a, b\}$, the

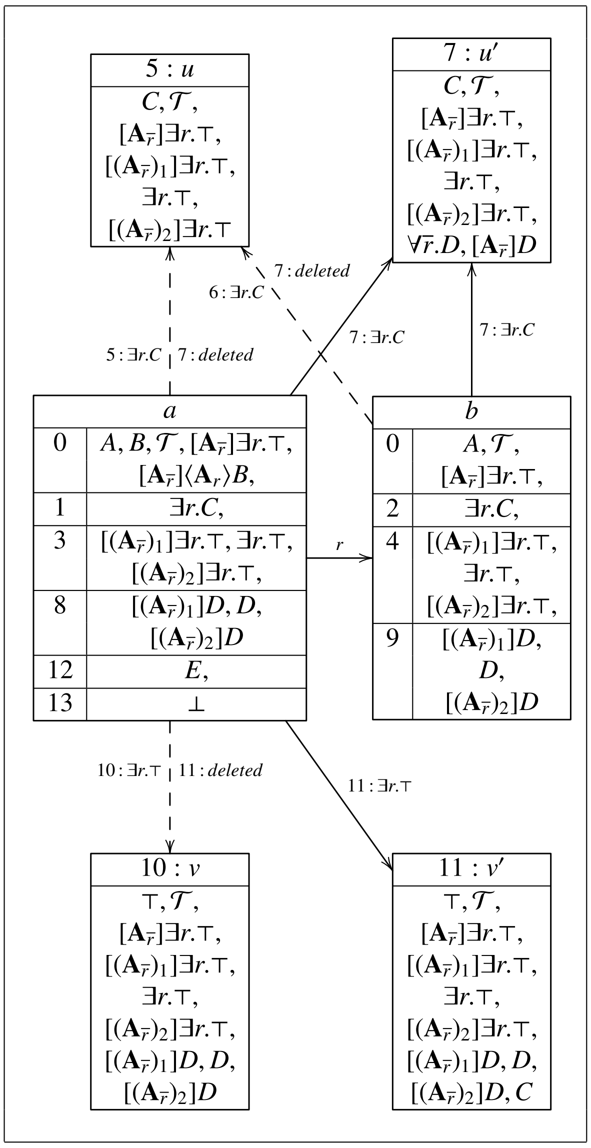
$5 : u$
$C, \mathcal{T},$ $[\mathbf{A}_{\overline{r}}]\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_1]\exists r.\top,$ $\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_2]\exists r.\top$
$7 : u'$
$C, \mathcal{T},$ $[\mathbf{A}_{\overline{r}}]\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_1]\exists r.\top,$ $\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_2]\exists r.\top,$ $\forall \overline{r}.D, [\mathbf{A}_{\overline{r}}]D$
$7 : deleted$
$6 : \exists r.C$
$7 : \exists r.C$
$7 : \exists r.C$
$5 : \exists r.C$
$7 : deleted$
a
0 $A, B, \mathcal{T}, [\mathbf{A}_{\overline{r}}]\exists r.\top,$ $[\mathbf{A}_{\overline{r}}]\langle \mathbf{A}_r \rangle B,$
1 $\exists r.C,$
3 $[(\mathbf{A}_{\overline{r}})_1]\exists r.\top, \exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_2]\exists r.\top,$
8 $[(\mathbf{A}_{\overline{r}})_1]D, D,$ $[(\mathbf{A}_{\overline{r}})_2]D$
12 $E,$
13 $\bot$
r
b
0 $A, \mathcal{T},$ $[\mathbf{A}_{\overline{r}}]\exists r.\top,$
2 $\exists r.C,$
4 $[(\mathbf{A}_{\overline{r}})_1]\exists r.\top,$ $\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_2]\exists r.\top,$
9 $[(\mathbf{A}_{\overline{r}})_1]D,$ $D,$ $[(\mathbf{A}_{\overline{r}})_2]D$
$10 : \exists r.\top$
$11 : deleted$
$11 : \exists r.\top$
$10 : v$
$\top, \mathcal{T},$ $[\mathbf{A}_{\overline{r}}]\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_1]\exists r.\top,$ $\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_2]\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_1]D, D,$ $[(\mathbf{A}_{\overline{r}})_2]D$
$11 : v'$
$\top, \mathcal{T},$ $[\mathbf{A}_{\overline{r}}]\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_1]\exists r.\top,$ $\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_2]\exists r.\top,$ $[(\mathbf{A}_{\overline{r}})_1]D, D,$ $[(\mathbf{A}_{\overline{r}})_2]D, C$

Figure 1. An illustration for Example 3.

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 : *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 (1).
2: Applying ($\sqsubseteq$) to $x = b$ using the clause (1).
3: Applying ($\forall_I$) to the nodes $x = a$ and $y = b$.
4: Applying ($\forall_i$) to the nodes a and b.
5: Applying ($\exists$) to $x = a$ and the concept $\exists r.C$.
6: Applying ($\exists$) to $x = b$ and the concept $\exists r.C$.
7: Applying ($\sqsubseteq$) to $x = u$ using the clause (2).
8: Applying ($\forall_I$) to the nodes $x = a$ and $y = u'$.
9: Applying ($\forall_I$) to the nodes $x = b$ and $y = u'$.
10: Applying ($\exists$) to $x = a$ and the concept $\exists r.\top$.
11: Applying ($\sqsubseteq$) to $x = v$ using the clause (3).
12: Applying ($\sqsubseteq$) to $x = a$ using the clause (4).
13: Applying ($\sqsubseteq$) to $x = a$ using the clause (6).

Since $\bot$ was added to *Label*(a), Algorithm 1 returns *false*, and by Corollary 9, the knowledge base *KB* is unsatisfiable.

5. Proofs

Define $closure_{\mathbf{A}}(\mathcal{T})$ to be the smallest set of formulas 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 $closure_{\mathbf{A}}(\mathcal{T})$,
- 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 $q \in Q_{\mathsf{A}}$ then $[\mathsf{A}_q]C \in closure_{\mathbf{A}}(\mathcal{T})$,
- $\{[\mathbf{A}_{\overline{R}}]\exists R.\top \mid R \in \mathbf{R}\} \subseteq closure_{\mathbf{A}}(\mathcal{T})$,
- if $A \in closure_{\mathbf{A}}(\mathcal{T})$ and $R \in \mathbf{R}$ then $[\mathbf{A}_{\overline{R}}]\langle \mathbf{A}_R \rangle A \in closure_{\mathbf{A}}(\mathcal{T})$.

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

Lemma 6. Algorithm 1 runs in polynomial time in the size of $\mathcal{A}$ (when assuming that $\mathcal{R}$ 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 `Find`(X) for $X \subseteq closure_{\mathbf{A}}(\mathcal{T})$ runs in constant time,
- procedure `CheckPremise`(x, C) runs in $O(n)$ steps (C does not depend on $\mathcal{A}$),
- 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 into Δ,
3. some $Next(x, \exists R.C)$ is defined the first time to be some $y \in \Delta \setminus \Delta_0$,
4. some $Next(x, \exists R.C)$ 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 $O(n^3)$. $\lhd$

Lemma 7. If Algorithm 1 returns *true* then the knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ is satisfiable.

Proof: Suppose Algorithm 1 returns *true* for $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$. We will refer to the data structures used by that run of Algorithm 1. A model for $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ 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 counter parts of Δ and $Edges$, respectively, together with a mapping $f : \Delta' \to \Delta$ and a queue *unresolved* of elements of Δ' as follows:

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

The resulting data structures can be infinite. Let $\mathcal{I}$ be the interpretation with $\Delta^{\mathcal{I}} = \Delta'$, specified by:

- 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 $\langle u, R, v \rangle \in Edges'$ then $\langle u, v \rangle \in R^{\mathcal{I}}$,
  - for every word $S_1 \ldots S_k$ accepted by $\mathbf{A}_R$, $S_1^{\mathcal{I}} \circ \cdots \circ S_k^{\mathcal{I}} \subseteq R^{\mathcal{I}}$.

We show that $\mathcal{I}$ is a model of $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$. For this it suffices to prove that, for every $u \in \Delta'$ and every $\varphi \in Label(f(u))$, $u \in \varphi^{\mathcal{I}}$. 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 $\varphi = \exists R.C$: Since $\varphi \in Label(f(u))$, there exists $v \in \Delta^{\mathcal{I}}$ such that $\langle u, v \rangle \in R^{\mathcal{I}}$ and $Next(f(u), \exists R.C) = f(v)$. We have that $C \in Label(f(v))$. By the inductive assumption, $v \in C^{\mathcal{I}}$, and hence $u \in \varphi^{\mathcal{I}}$.
- Case $\varphi = \forall R.A$: Let v be any element of $\Delta^{\mathcal{I}}$ such that $\langle u, v \rangle \in R^{\mathcal{I}}$. We show that $v \in A^{\mathcal{I}}$. Since $\langle u, v \rangle \in R^{\mathcal{I}}$, there exist a word $S_1 \ldots S_k$ accepted by $\mathbf{A}_R$ and elements $u_0 = u, u_1, \ldots, u_{k-1}, u_k = v$ such that, for every $1 \leq i \leq k$, $\langle u_{i-1}, u_i \rangle \in S_i^{\mathcal{I}}$, 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 $S_1 \ldots S_k$ is accepted by A, there exist states $q_0 = q_{\mathsf{A}}, q_1, \ldots, q_k$ such that $q_k \in F_{\mathsf{A}}$ and $\langle q_{i-1}, S_i, q_i \rangle \in \delta_{\mathsf{A}}$ for every $1 \leq i \leq k$. 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, since $\langle u_{i-1}, S_i, u_i \rangle \in Edges'$ or $\langle u_i, \overline{S_i}, u_{i-1} \rangle \in Edges'$, 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, by the inductive assumption, $v \in A^{\mathcal{I}}$.
- 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 `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 $A \in Label(f(u))$.
- Case $C_i = \exists R.A$: Since $u \in C_i^{\mathcal{I}}$, there exist a word $S_1 \dots S_k$ accepted by $\mathbf{A}_R$ and elements $u_0 = u, u_1, \dots, u_{k-1}, u_k$ such that $u_k \in A^{\mathcal{I}}$ and, for every $1 \le i \le k$, $\langle u_{i-1}, u_i\rangle \in S_i^{\mathcal{I}}$, 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 $\mathsf{A} = \mathbf{A}_{\overline{R}}$. Since $\overline{S}_k \dots \overline{S}_1$ is accepted by A, there exist states $q_k = q_{\mathsf{A}}, q_{k-1}, \dots, q_0$ such that $q_0 \in F_{\mathsf{A}}$ and $\langle q_i, \overline{S}_i, q_{i-1}\rangle \in \delta_{\mathsf{A}}$ for every $k \ge i \ge 1$. Since $u_k \in A^{\mathcal{I}}$, we have that $A \in Label(f(u_k))$ and, by saturation, $[\mathbf{A}_{\overline{R}}]\langle \mathbf{A}_R\rangle A \in Label(f(u_k))$, which means $[\mathsf{A}_{q_k}]\langle \mathbf{A}_R\rangle A \in Label(f(u_k))$. For each i from k to 1, since $\langle u_i, \overline{S}_i, u_{i-1}\rangle \in Edges'$ or $\langle u_{i-1}, S_i, u_i\rangle \in Edges'$, it follows that $[\mathsf{A}_{q_{i-1}}]\langle \mathbf{A}_R\rangle A \in Label(f(u_{i-1}))$. Since $q_0 \in F_{\mathsf{A}}$ and $u_0 = u$, it follows that $\langle \mathbf{A}_R\rangle A \in Label(f(u))$.
- Case $C_i = \forall\exists R.A$: Since $u \in C_i^{\mathcal{I}}$, we have that $u \in (\forall R.A)^{\mathcal{I}}$ and $u \in (\exists R.\top)^{\mathcal{I}}$. Thus, there exist a word $S_1 \dots S_k$ accepted by $\mathbf{A}_R$ and elements $u_0 = u, u_1, \dots, u_{k-1}, u_k$ such that, for every $1 \le i \le k$, $\langle u_{i-1}, u_i\rangle \in S_i^{\mathcal{I}}$, 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 $\mathsf{A} = \mathbf{A}_{\overline{R}}$. Since $\overline{S}_k \dots \overline{S}_1$ is accepted by A, there exist states $q_k = q_{\mathsf{A}}, q_{k-1}, \dots, q_0$ such that $q_0 \in F_{\mathsf{A}}$ and $\langle q_i, \overline{S}_i, q_{i-1}\rangle \in \delta_{\mathsf{A}}$ for every $k \ge i \ge 1$. By saturation, $[\mathbf{A}_{\overline{R}}]\exists R.\top \in Label(f(u_k))$, which means $[\mathsf{A}_{q_k}]\exists R.\top \in Label(f(u_k))$. For each i from k to 1, since $\langle u_i, \overline{S}_i, u_{i-1}\rangle \in Edges'$ or $\langle u_{i-1}, S_i, u_i\rangle \in Edges'$, it follows that $[\mathsf{A}_{q_{i-1}}]\exists R.\top \in Label(f(u_{i-1}))$. Since $q_0 \in F_{\mathsf{A}}$ and $u_0 = u$, it follows that $\exists R.\top \in Label(f(u))$. Therefore, $Next(f(u), \exists R.\top)$ is defined and there exists $v' \in \Delta^{\mathcal{I}}$ with $f(v') = Next(f(u), \exists R.\top)$. We have that $\langle u, v'\rangle \in R^{\mathcal{I}}$. Since $u \in (\forall R.A)^{\mathcal{I}}$, it follows that $v' \in A^{\mathcal{I}}$ and hence $A \in Label(f(v'))$, which means $A \in Label(Next(f(u), \exists R.\top))$.

We have shown that `CheckPremise`$(f(u), C)$ returned *true*. It follows that $D \in Label(f(u))$, 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 7 given above is in fact an assertion about the completeness of the procedure. It remains to show the soundness: if $\bot$ is added to $Label(x)$ for some $x \in \Delta$ (which causes the algorithm to return *false*), then the knowledge base $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ is unsatisfiable. 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* is satisfiable then Algorithm 1 returns *true* for it. By assuming that *KB* is satisfiable and using any fixed model $\mathcal{I}$ of *KB*, 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 8. Let $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ be a clausal Horn-$\mathcal{R}eg^I$ knowledge base. Suppose *KB* is satisfiable and $\mathcal{I}$ is a model of *KB*. Consider an execution of Algorithm 1 for *KB* and any moment after executing the step 7 of that execution. Let $r = \{\langle x, u\rangle \in \Delta \times \Delta^{\mathcal{I}} \mid u \in (Label(x))^{\mathcal{I}}\}$. Then:

1. for every $a \in \mathbf{I}$ occurring in $\mathcal{A}$, $r(a, a^{\mathcal{I}})$ holds;
2. for every $x, y \in \Delta$, $u, v \in \Delta^{\mathcal{I}}$ and $\exists R.C$ such that $Next(x, \exists R.C) = y$, if $r(x, u)$ holds, $R^{\mathcal{I}}(u, v)$ holds and $v \in C^{\mathcal{I}}$, then $r(y, v)$ holds;
3. for every $x \in \Delta$, there exists $u \in \Delta^{\mathcal{I}}$ such that $r(x, u)$ holds.

Note that if $r(x, u)$ holds then $u \in (Label(x))^{\mathcal{I}}$, which means $Label(x)$ is satisfied at (and hence "justified by") u in $\mathcal{I}$. The second assertion of the lemma implies that if $Next(x, \exists R.\top) = y$, $r(x, u)$ and $R^{\mathcal{I}}(u, v)$ hold then $r(y, v)$ holds. The first two assertions of this lemma can be proved by induction on the number of executed steps in a way similar to the proof of [24, Lemma 3.5]. The last assertion follows from the previous ones, because every $x \in \Delta$ is/was at some step reachable from Δ_0 and $Label(x)$ was never changed.

Corollary 9. If $KB = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}\rangle$ is a satisfiable clausal Horn-$\mathcal{R}eg^I$ knowledge base then Algorithm 1 returns *true* for it.

Proof sketch: By the last assertion of Lemma 8, $\bot$ was never added to $Label(x)$ for any $x \in \Delta$. This means that Algorithm 1 does not return *false*. As it always terminates (by Lemma 6), it must return *true*. ◁

6. Conclusions and Future Work

We have explained our technique of dealing with non-seriality that leads to a solution for the important issue of allowing the concept constructor $\forall\exists R.C$ to appear at the left hand side of $\sqsubseteq$ in terminological inclusion axioms. We have developed an algorithm with PTIME data complexity for checking satisfiability of Horn-$\mathcal{R}eg^I$ knowledge bases. This shows that both Horn-$\mathcal{R}eg$ and Horn-$\mathcal{R}eg^I$ have PTIME data complexity, solving an open problem of [12].

Recently, in [26] we have introduced Horn-DL as a generalization of Horn-$\mathcal{R}eg^I$ that still has PTIME data complexity. The full manuscript on Horn-DL [27] is to be improved and not published yet. As future work, we intend to develop efficient methods for evaluating queries to Horn-$\mathcal{R}eg^I$ and Horn-DL knowledge bases. As Horn-$\mathcal{R}eg^I$ is a restricted version of Horn-DL, we expect to have more optimization techniques for query evaluation in Horn-$\mathcal{R}eg^I$.

Acknowledgements

This work was supported by the Polish National Science Centre (NCN) under Grants No. 2011/01/B/ST6/02759 and 2011/01/B/ST6/02769. We would also like to thank the anonymous reviewers for helpful comments and suggestions.

References

[1] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible SROIQ, in: P. Doherty, J. Mylopoulos, C. Welty (Eds.), Proceedings of KR'2006, AAAI Press, 2006, pp. 57–67.

[2] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope., in: L. Kaelbling, A. Saffiotti (Eds.), Proceedings of IJCAI'2005, Morgan-Kaufmann Publishers, 2005, pp. 364–369.

[3] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope further, in: Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions, 2008.

[4] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The DL-Lite family, J. Autom. Reasoning 39 (3) (2007) 385–429.

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

[6] B. Grosof, I. Horrocks, R. Volz, S. Decker, Description logic programs: combining logic programs with description logic., in: Proceedings of WWW'2003, 2003, pp. 48–57.

[7] U. Hustadt, B. Motik, U. Sattler, Reasoning in description logics by a reduction to disjunctive Datalog, J. Autom. Reasoning 39 (3) (2007) 351–384.

[8] A. Krisnadhi, C. Lutz, Data complexity in the EL family of description logics, in: N. Dershowitz, A. Voronkov (Eds.), Proceedings of LPAR'2007, LNCS 4790, Springer, 2007, pp. 333–347.

[9] M. Krötzsch, S. Rudolph, P. Hitzler, Conjunctive queries for a tractable fragment of OWL 1.1, in: Proceedings of ISWC'2007 + ASWC'2007, LNCS 4825, Springer, 2007, pp. 310–323.

[10] R. Rosati, On conjunctive query answering in $\mathcal{EL}$, in: Proceedings of DL'2007, pp. 451–458.

[11] T. Eiter, G. Gottlob, M. Ortiz, M. Simkus, Query answering in the description logic horn-shiq, in: Proceedings of JELIA'2008, Vol. 5293 of LNCS, Springer, 2008, pp. 166–179.

[12] L. Nguyen, Horn knowledge bases in regular description logics with PTime data complexity, Fundamenta Informaticae 104 (4) (2010) 349–384.

[13] M. Ortiz, S. Rudolph, M. Simkus, Query answering in the Horn fragments of the description logics SHOIQ and SROIQ, in: T. Walsh (Ed.), Proceedings of IJCAI 2011, 2011, pp. 1039–1044.

[14] M. Krötzsch, S. Rudolph, P. Hitzler, Complexity boundaries for Horn description logics, in: Proceedings of AAAI'2007, AAAI Press, 2007, pp. 452–457.

[15] S. Brandt, Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else?, in: R. de Mántaras, L. Saitta (Eds.), Proceedings of ECAI'2004, IOS Press, 2004, pp. 298–302.

[16] L. Nguyen, A bottom-up method for the deterministic Horn fragment of the description logic $\mathcal{ALC}$, in: M. F. et al. (Ed.), Proceedings of JELIA 2006, LNAI 4160, Springer-Verlag, 2006, pp. 346–358.

[17] L. Nguyen, T.-B.-L. Nguyen, A. Szałas, On Horn knowledge bases in regular description logic with inverse, in: Proceedings of KSE'2013 (Part I. AISC), Advances in Intelligent and Soft Computing, Springer, 2014, pp. 37–50.

[18] S. Demri, The complexity of regularity in grammar logics and related modal logics., Journal of Logic and Computation 11 (6) (2001) 933–960.

[19] S. Demri, H. de Nivelle, Deciding regular grammar logics with converse through first-order logic, Journal of Logic, Language and Information 14 (3) (2005) 289–329.

[20] R. Goré, L. Nguyen, A tableau system with automaton-labelled formulae for regular grammar logics, in: B. Beckert (Ed.), Proceedings of TABLEAUX 2005, LNAI 3702, Springer-Verlag, 2005, pp. 138–152.

[21] L. Nguyen, A. Szałas, ExpTime tableau decision procedures for regular grammar logics with converse, Studia Logica 98 (3) (2011) 387–428.

[22] L. Nguyen, A. Szałas, On the Horn fragments of serial regular grammar logics with converse, in: Proceedings of KES-AMSTA'2013, Vol. 252 of Frontiers of Artificial Intelligence and Applications, IOS Press, 2013, pp. 225–234.

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

[24] L. Nguyen, Constructing finite least Kripke models for positive logic programs in serial regular grammar logics, Logic Journal of the IGPL 16 (2) (2008) 175–193.

[25] B. Dunin-Kęplicz, L. Nguyen, A. Szałas, Tractable approximate knowledge fusion using the Horn fragment of serial propositional dynamic logic, Int. J. Approx. Reasoning 51 (3) (2010) 346–362.

[26] L. Nguyen, T.-B.-L. Nguyen, A. Szałas, Horn-DL: An expressive Horn description logic with PTime data complexity, in: W. Faber, D. Lembo (Eds.), Proceedings of RR'2013, Vol. 7994 of LNCS, Springer, 2013, pp. 259–264.

[27] L. Nguyen, T.-B.-L. Nguyen, A. Szałas, A long version of the paper [26], Available at `http://www.mimuw.edu.pl/~nguyen/horn_dl_long.pdf`.