Horn Knowledge Bases in Regular Description Logics with PTIME Data Complexity *

Linh Anh Nguyen

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

May 2007 (last revised: November 2010)

Abstract. Developing a good formalism and an efficient decision procedure for the *instance checking problem* is desirable for practical application of description logics. The data complexity of the instance checking problem is coNP-complete even for *Horn knowledge bases* in the basic description logic $\mathcal{ALC}$. In this paper, we present and study *weakenings* with PTIME data complexity of the instance checking problem for Horn knowledge bases in *regular description logics*. We also study cases when the weakenings are an exact approximation. In contrast to previous related work of other authors, our approach deals with the case when the constructor $\forall$ is allowed in *premises* of *program clauses* that are used as terminological axioms.

1 Introduction

Description logics (DLs) are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way [2]. We can use them, for example, for conceptual modeling or as ontology languages. The logical formalisms of OWL (Web Ontology Language), recommended by W3C, are based on description logics.

Description logics represent the domain of interest in terms of concepts, individuals, and roles. A knowledge base in a DL is a tuple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ consisting of an RBox $\mathcal{R}$ of role axioms, a TBox $\mathcal{T}$ of terminological axioms, and an ABox $\mathcal{A}$ of facts about individuals (objects) and roles. The instance checking problem in a DL is to check whether a given individual a is an instance of a given concept C w.r.t. a knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, which is written as $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$. This problem in the basic description logic $\mathcal{ALC}$ (with $\mathcal{R} = \emptyset$) is EXPTIME-complete [39]. From the point of view of deductive databases, $\mathcal{A}$ is assumed to be much larger than $\mathcal{R}$ and $\mathcal{T}$, and it makes sense to consider the data complexity, which is measured when the query specified by $\mathcal{R}$, $\mathcal{T}$, C and a is fixed while $\mathcal{A}$ varies as input data. In [20], Hustadt et al. proved that the data complexity of the instance checking problem in the description logic $\mathcal{SHIQ}$, which extends $\mathcal{ALC}$ with transitive roles, role hierarchies, inverse roles and number restrictions, is coNP-complete.

It is interesting from both the theoretical and practical points of view to find and study fragments of DLs with tractable (PTIME) data complexity. In this work, we consider two problems:

* This is a revised and extended version of our papers [32, 33]. It will appear in Fundamenta Informaticae. The work has been partially supported by grant N N206 3982 33 from the Polish Ministry of Science and Higher Education.

- The first problem is to identify classes of $\mathcal{R}$, $\mathcal{T}$, $\mathcal{A}$, C such that the instance checking problem $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$ has PTIME data complexity.
- Using a restricted language to specify a class of allowed $\mathcal{R}$, $\mathcal{T}$, $\mathcal{A}$, C, let $InstanceChecking(\mathcal{R}, \mathcal{T}, \mathcal{A})$ denote the set of $C(a)$ such that $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$.

  The second problem is to approximate the set $InstanceChecking(\mathcal{R}, \mathcal{T}, \mathcal{A})$ by a subset $\mathbf{P}(\mathcal{R}, \mathcal{T}, \mathcal{A})$ such that checking whether $C(a) \in \mathbf{P}(\mathcal{R}, \mathcal{T}, \mathcal{A})$ can be done in polynomial time in the size of $\mathcal{A}$.

  Checking whether $C(a) \in \mathbf{P}(\mathcal{R}, \mathcal{T}, \mathcal{A})$ is then called a *weakening* of the checking $C(a) \in InstanceChecking(\mathcal{R}, \mathcal{T}, \mathcal{A})$. For short, in that case we also say that $\mathbf{P}$ is a weakening of $InstanceChecking$. Amongst two weakenings $\mathbf{P}$ and $\mathbf{P}'$ of $InstanceChecking$, the first one is *better* than the second one if $\mathbf{P}(\mathcal{R}, \mathcal{T}, \mathcal{A}) \supseteq \mathbf{P}'(\mathcal{R}, \mathcal{T}, \mathcal{A})$ for every $\mathcal{R}$, $\mathcal{T}$, $\mathcal{A}$ allowed by the restricted language.

A natural approach to the mentioned problems is to consider Horn fragments of DLs. Several researchers have introduced and studied different Horn fragments of DLs [16, 4, 1, 20, 5, 22, 24, 37]. In this work, we define and study the *general Horn fragment* of DLs that extend $\mathcal{ALC}$ with a *regular RBox*.

This paper is structured as follows. In Section 2 we recall notation and semantics of the description logic $\mathcal{ALC}$, and define regular DLs and their general Horn fragment. Having basic definitions, in Section 3 we give an overview of the results of this work. The formulations of our results in the overview are introductory. They use a few notions that are formally defined in the latter sections. In Section 4 we discuss related work. In Section 5 we present several examples demonstrating the potential of the general Horn fragment of regular DLs in real-world applications. The technical part of this work consists of Sections 6, 7 and 8. In Section 6 we introduce pseudo-interpretations. In Sections 7 we present an algorithm for constructing a least pseudo-model of a given deterministic Horn knowledge base in a given regular DL and analyze its data complexity. In Section 8 we provide characterizations of such least pseudo-models and present related results involving the instance checking problem. Concluding remarks are given in Section 9.

2 Preliminaries

2.1 Notation and Semantics of $\mathcal{ALC}$

The basic description logic $\mathcal{ALC}$ is a notational variant of the multimodal logic $K_{(n)}$. The notations change as follows: primitive propositions are called *atomic concepts*, the connectives $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ are written respectively as $\sqcap$, $\sqcup$, $\sqsubseteq$, $\doteq$, and formulas of the form $\Box_i\varphi$ or $\Diamond_i\varphi$ are written respectively as $\forall R_i.\varphi$ and $\exists R_i.\varphi$. In the DL literature, $\neg$, $\sqcap$, $\sqcup$, $\forall$, $\exists$ are traditionally called concept constructors, while $\sqsubseteq$ and $\doteq$ are used only in *terminological axioms* and cannot be nested. On the one hand, it is intuitive to not call a *general concept inclusion* $C \sqsubseteq D$ a concept. But on the other hand, if "concept" is understood as a set of objects (in an interpretation), then $C \sqsubseteq D$ can be treated as a "concept". Besides, one may prefer to shorten an axiom of the form $\top \sqsubseteq C$ to C and treat it as a global assumption. In this work, instead of the terms "concept", "terminological axiom", "general concept inclusion", we tend to use only one term "formula".

Definition 2.1. In the *primitive language* of $\mathcal{ALC}$, *formulas* are defined using the following BNF grammar:

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

where A denotes an *atomic concept* and R denotes a *role name*. ◁

By this definition, a formula may contain nested occurrences of $\sqsubseteq$ and $\doteq$. We will use letters like A, B to denote atomic concepts, and letters like C, D to denote formulas.

Definition 2.2. An *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$, the *interpretation function* of $\mathcal{I}$, that maps every atomic concept to a subset of $\Delta^{\mathcal{I}}$ and every role name to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. ◁

The interpretation function is extended to interpret every formula as follows:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(C \sqsubseteq D)^{\mathcal{I}} &= (\neg C \sqcup D)^{\mathcal{I}} \\
(C \doteq D)^{\mathcal{I}} &= ((C \sqsubseteq D) \sqcap (D \sqsubseteq C))^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y.(x,y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}.
\end{aligned}$$

Using the notations of modal logic, where $\Delta^{\mathcal{I}}$ is treated as a set of possible worlds and $R^{\mathcal{I}}$ is treated as an accessibility relation, we also write $\mathcal{I}, x \models C$ to denote $x \in C^{\mathcal{I}}$.

Definition 2.3. A *TBox* is a finite set of formulas. An interpretation $\mathcal{I}$ *validates* a formula C if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model of a TBox* $\mathcal{T}$ if $\mathcal{I}$ validates all $C \in \mathcal{T}$. ◁

We use letters like a and b to denote *individuals*.

Definition 2.4. An *ABox* is a finite set of *assertions* of the form $A(a)$ (*concept assertion*) or $R(a,b)$ (*role assertion*). An interpretation $\mathcal{I}$, which additionally maps every individual a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, is a *model of an ABox* $\mathcal{A}$ iff $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$) holds for every assertion $A(a)$ (resp. $R(a,b)$) of $\mathcal{A}$. ◁

In [20], an ABox as defined above is said to be *extensionally reduced*.
We will write $\mathcal{I} \models C(a)$ to denote $\mathcal{I}, a^{\mathcal{I}} \models C$ (which means $a^{\mathcal{I}} \in C^{\mathcal{I}}$).

2.2 Regular Description Logics

Definition 2.5. A (context-free) *RBox* is a finite set of *role axioms* of the form

$$R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$$

where $k \geq 0$ and the left hand side stands for the identity relation if $k = 0$ [19, 1, 20]. An interpretation $\mathcal{I}$ is a *model of an RBox* $\mathcal{R}$ iff $R^{\mathcal{I}}_{s_1} \circ \ldots \circ R^{\mathcal{I}}_{s_k} \subseteq R^{\mathcal{I}}_t$ for every role axiom $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ of $\mathcal{R}$. ◁

We say that $R_{s_1} \circ \cdots \circ R_{s_k} \sqsubseteq R_t$ is a *consequence of an RBox* $\mathcal{R}$ if every model $\mathcal{I}$ of $\mathcal{R}$ is also a model of the role axiom (i.e. $R^{\mathcal{I}}_{s_1} \circ \ldots \circ R^{\mathcal{I}}_{s_k} \subseteq R^{\mathcal{I}}_t$).

Definition 2.6. A *knowledge base* is a triple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ consisting of an RBox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We say that $\mathcal{I}$ is a *model of a knowledge base* $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ if it is a model of all $\mathcal{R}$, $\mathcal{T}$, $\mathcal{A}$. An individual a is an *instance* of a formula C (understood as a "concept") w.r.t. $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, write $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$, if $\mathcal{I} \models C(a)$ holds for every model $\mathcal{I}$ of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. ◁

From now on, we assume that role names are of the form R_t for $t \in \mathrm{IND}$, where IND is a fixed finite set of indices. A role axiom $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ corresponds to the grammar rule $t \to s_1 \ldots s_k$. An RBox $\mathcal{R}$ thus corresponds to a semi-Thue system, denoted by $G(\mathcal{R})$, which is like a context free grammar, but it has no designated start symbol and there is no distinction between terminal and non-terminal symbols.

Definition 2.7. If for every $t \in \mathrm{IND}$, the set of words over alphabet IND that are derivable from t using $G(\mathcal{R})$ is a regular language and recognized by a finite automaton $\mathbf{A}_t$, then we call $\mathcal{R}$ a *regular RBox* and $(\mathbf{A}_t)_{t \in \mathrm{IND}}$ the *automata specifying* $\mathcal{R}$. ◁

We assume that the automata specifying $\mathcal{R}$ are given explicitly because checking whether a context-free grammar generates a regular language is undecidable (see, e.g., [25]). (In [8], a "regular grammar logic" is specified either by a left/right linear grammar or by finite automata. Note that every left/right linear grammar can be transformed in polynomial time to an equivalent finite automaton.)

Recall that a *finite automaton* $\mathbf{A}$ is a tuple $\langle \Sigma, Q, I, \delta, F \rangle$, where Σ is the alphabet (in our case, $\Sigma = \mathrm{IND}$), Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. A *run* of $\mathbf{A}$ on a word $s_1 \ldots s_k$ is a finite sequence of states $q_0, q_1, \ldots, q_k$ such that $q_0 \in I$ and $\delta(q_{i-1}, s_i, q_i)$ holds for every $1 \leq i \leq k$. It is an *accepting run* if $q_k \in F$. We say that $\mathbf{A}$ *accepts* word w if there exists an accepting run of $\mathbf{A}$ on w. The set of all words accepted/recognized by $\mathbf{A}$ is denoted by $\mathcal{L}(\mathbf{A})$.

Definition 2.8. By $\mathcal{R}eg$ we denote the class of description logics that extend $\mathcal{ALC}$ with a regular RBox. Each regular RBox identifies a logic of $\mathcal{R}eg$, which is called a *regular ($\mathcal{R}eg$) DL*. We sometimes use $\mathcal{R}eg$ also to refer to an arbitrary regular DL. ◁

2.3 The General Horn Fragment of $\mathcal{R}eg$

Definition 2.9.

- A formula C is called a *positive formula* if it does not contain $\neg$, $\sqsubseteq$, $\doteq$, and is called a *negative formula* if it is $\neg D$ for some positive formula D.
- *Non-negative Horn formulas* are defined by the following BNF grammar, where A denotes an atomic concept and D denotes a positive formula:

$$C ::= \top \mid A \mid D \sqsubseteq C \mid C \sqcap C \mid \forall R_t.C \mid \exists R_t.C$$

- A *Horn formula* is either a non-negative Horn formula or a negative formula.

- A *program clause* is a non-negative Horn formula.
- A *positive logic program* is a finite set of program clauses.
- A knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ is called a *Horn knowledge base* in $\mathcal{R}eg$ if $\mathcal{R}$ is a regular RBox and $\mathcal{T}$ is a positive logic program.[1] ◁

Definition 2.10. A *premise* of a program clause C is a formula D such that $D \sqsubseteq D'$ is a subformula of C for some D'. A *conclusion* of a program clause C is a formula D *without* $\sqsubseteq$ such that either $D = C$ or $D' \sqsubseteq D$ is a subformula of C for some D'. ◁

For example, the program clause $(A \sqcap B) \sqsubseteq \exists R.((A' \sqsubseteq A'') \sqcap (B' \sqsubseteq B''))$ has premises $A \sqcap B$, A', B', and conclusions A'', B''.

Concerning the instance checking problem $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$, the *general Horn fragment of* $\mathcal{R}eg$ consists of positive logic programs used for $\mathcal{T}$ and positive formulas used for C and regular RBoxes used for $\mathcal{R}$.

3 An Overview of the Results of This Work

In DLs roles are not required to satisfy the seriality condition $\forall x \exists y R_i(x, y)$. As shown in [27, 28], non-seriality causes a problem of nondeterminism for modal logics.[2] For the same reason, the data complexity of the instance checking problem for the general Horn fragment of $\mathcal{ALC}$ and $\mathcal{R}eg$ is coNP-hard (see Appendix A for the proof).

In this work, we first study approximating the instance checking problem $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$ for a Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ in $\mathcal{R}eg$ and a positive formula C by a weakening $\mathbf{P}$ such that checking whether $C(a) \in \mathbf{P}(\mathcal{R}, \mathcal{T}, \mathcal{A})$ can be done in polynomial time in the size of $\mathcal{A}$. We then identify the cases when the approximation is exact.

We extend the language with the constructor $\forall\exists$, which creates a formula $\forall\exists R_t.C$ from a role name R_t and a formula C. We provide two semantics for dealing with $\forall\exists R_t.C$, where the second one depends on a given RBox $\mathcal{R}$.

Definition 3.1. Let

$$\mathsf{Sem}_1(\forall\exists R_t.C) = \{\forall R_t.C, \exists R_t.\top\}$$

$$\mathsf{Sem}_{2,\mathcal{R}}(\forall\exists R_t.C) = \{\forall R_t.C\} \cup \{\forall R_{s_1} \ldots \forall R_{s_{i-1}} \exists R_{s_i}.\top \mid R_{s_1} \circ \cdots \circ R_{s_k} \sqsubseteq R_t \text{ is a consequence of } \mathcal{R} \text{ and } 1 \le i \le k\}.$$

Given a model $\mathcal{I}$ of an RBox $\mathcal{R}$, $x \in \Delta^{\mathcal{I}}$ and $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$, define $\mathcal{I}, x \models_{\mathfrak{s}} C$

- in the usual way (as in modal logic) if C is not of the form $\forall\exists R_t.D$, and that
- $\mathcal{I}, x \models_{\mathfrak{s}} \forall\exists R_t.D$ if $\mathcal{I}, x \models_{\mathfrak{s}} D'$ for every $D' \in \mathfrak{s}(\forall\exists R_t.D)$. ◁

Note that $\mathsf{Sem}_{2,\mathcal{R}}(\forall\exists R_t.C)$ may be an infinite set.

From now on, by $\mathfrak{s}$ we denote either Sem_1 or $\mathsf{Sem}_{2,\mathcal{R}}$.

[1] Note that negative formulas are not allowed in a Horn knowledge base because they can play only the role of constraints to detect inconsistency of the knowledge base itself.

[2] Every positive logic program has a least KD-model, but may not have any least K-model [27]. The complexity of checking satisfiability of a set of modal Horn formulas with modal depth bounded by $k \ge 2$ in KD is PTIME-complete [27] and in K is NP-complete [28].

Example 3.2. If $\mathcal{R}$ is empty (the case of $\mathcal{ALC}$) then $R_t \sqsubseteq R_t$ for $t \in \mathrm{IND}$ are the only consequences of $\mathcal{R}$ and $\mathsf{Sem}_{2,\mathcal{R}}(\forall\exists R_t.C) = \mathsf{Sem}_1(\forall\exists R_t.C) \equiv \{\forall R_t.C,\ \exists R_t.C\}$. If $\mathcal{R}$ consists only of role axioms of the form $R_s \circ R_s \sqsubseteq R_s$ and $R_t \circ R_t \sqsubseteq R_t$ is one of them, then $\mathsf{Sem}_{2,\mathcal{R}}(\forall\exists R_t.C) \equiv \{\forall R_t.C, \exists R_t.\top, \forall R_t.\exists R_t.\top\}$.

Note: The notions of "program clause", "positive logic program" and "Horn knowledge base" will always be understood as the ones defined in the primitive language (i.e. without the constructor $\forall\exists$).

Definition 3.3.

- A *positive formula* is still defined to be a formula without $\neg$, $\sqsubseteq$, $\doteq$, but from now on, we will distinguish three kinds of positive formulas: positive formulas (possibly with both $\forall$ and $\forall\exists$), positive formulas without $\forall\exists$, and positive formulas without $\forall$. Formulas of the last kind are called *positive allsome-formulas*. Note that they may contain $\exists$ and $\forall\exists$.
- If C is the formula obtained from a program clause C' by replacing every $\forall$ in the premises by $\forall\exists$ then we call C a *deterministic program clause* and the *deterministic version* of C'.
- A *deterministic positive logic program* is a finite set of deterministic program clauses. If a deterministic positive logic program $\mathcal{T}$ is obtained from a positive logic program $\mathcal{T}'$ by replacing every $\forall$ in the premises of the program clauses of $\mathcal{T}'$ by $\forall\exists$ then we call $\mathcal{T}$ the *deterministic version* of $\mathcal{T}'$.
- We call $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ a *deterministic Horn knowledge base* in $\mathcal{R}eg$ if $\mathcal{T}$ is the deterministic version of $\mathcal{T}'$ and $(\mathcal{R}, \mathcal{T}', \mathcal{A})$ is a Horn knowledge base in $\mathcal{R}eg$. ◁

Definition 3.4. We say that an interpretation $\mathcal{I}$ is an $\mathfrak{s}$-*model* of a deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ in $\mathcal{R}eg$ if $\mathcal{I}$ is a model of $\mathcal{R}$ and $\mathcal{A}$, and for every $x \in \Delta^{\mathcal{I}}$ and $C \in \mathcal{T}$, we have that $\mathcal{I}, x \models_{\mathfrak{s}} C$. We write $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathfrak{s}} C(a)$ to denote that, for every $\mathfrak{s}$-model $\mathcal{I}$ of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, it holds that $\mathcal{I}, a^{\mathcal{I}} \models_{\mathfrak{s}} C$. ◁

Definition 3.5. An interpretation $\mathcal{I}$ is said to be *less than or equal to* $\mathcal{I}'$ if for any positive formula C without $\forall\exists$ and for any individual a, $\mathcal{I} \models C(a)$ implies $\mathcal{I}' \models C(a)$. An interpretation $\mathcal{I}$ is said to be a *least model* of a Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ if $\mathcal{I}$ is a model $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ and $\mathcal{I}$ is less than or equal to any model $\mathcal{I}'$ of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. ◁

If $\mathcal{I}$ is a least model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ then, for any positive formula C without $\forall\exists$ and for any individual a, $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$ is equivalent to $\mathcal{I} \models C(a)$. Unfortunately, similarly to the case of the non-serial modal logic K, one cannot hope for the existence of such a model $\mathcal{I}$ in the general case. However, we can talk about *least* $\mathfrak{s}$*-pseudo-models* of *deterministic* Horn knowledge bases.

Note: The notions of *least* $\mathfrak{s}$*-pseudo-model* and $\mathcal{I} \models_{\mathfrak{s}} C(a)$ for an $\mathfrak{s}$*-pseudo-model* $\mathcal{I}$ will be defined in Section 6. We are now interested only in their properties related to the instance checking problem.

In this work, we present an algorithm that, given $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$ and a deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ in $\mathcal{R}eg$, constructs a finite least $\mathfrak{s}$-pseudo-model $\mathcal{I}$ of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. The algorithm runs in polynomial time and returns $\mathcal{I}$ with a polynomial size

in the size of $\mathcal{A}$. Moreover, checking $\mathcal{I} \models_{\mathfrak{s}} C(a)$ can be done in polynomial time in the size of $\mathcal{I}$ and C. Some important properties of the constructed structures are illustrated in Figure 1.

As shown by the first diagram of the figure, given a Horn knowledge base $(\mathcal{R}, \mathcal{T}', \mathcal{A})$ in $\mathcal{R}eg$ and a positive formula C without $\forall\exists$, for $\mathcal{T}$ being the deterministic version of $\mathcal{T}'$ and $\mathcal{I}_1$ (resp. $\mathcal{I}_2$) being a least Sem_1-pseudo-model (resp. $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model) of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, all of the four checking problems $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a)$, $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$, $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$, $\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$ are weakenings of the instance checking problem $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C(a)$. The first two weakenings cannot directly be decided, while the two remaining ones can be decided with PTIME data complexity. The weakening using $\mathcal{I}_1$ is better than the weakening using $\mathcal{I}_2$. We introduce $\mathsf{Sem}_{2,\mathcal{R}}$ and the related structure $\mathcal{I}_2$ due to the nice theoretical property that, for any positive allsome-formula C,

$$(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a) \Leftrightarrow \mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a).$$

For the case associated with the third diagram of the figure, we also have that

$$(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a) \Leftrightarrow \mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a).$$

Given a Horn knowledge base $(\mathcal{R}, \mathcal{T}', \mathcal{A})$ in $\mathcal{R}eg$ and a positive formula C without $\forall\exists$, for $\mathcal{T}$ being the deterministic version of $\mathcal{T}'$ and $\mathcal{I}_1$ being a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, the approximation of checking $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C(a)$ by checking $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$ is exact when, for every $t \in \mathrm{IND}$ (Corollary 8.5):

- either the constructor $\forall R_t.D$ does not occur in C and the premises of the program clauses of $\mathcal{T}'$
- or $\exists R_s.\top \in \mathcal{T}'$ for every $s \in \mathrm{IND}$ occurring in some word accepted by $\mathbf{A}_t$
- or $\mathcal{L}(\mathbf{A}_t) = \{t\}$ and the constructor $\forall R_t.D$ may occur in C and the premises of the program clauses of $\mathcal{T}'$ only in the form $(\forall R_t.D \sqcap \exists R_t.D')$ for some arbitrary D'.

4 Related Work

$\mathcal{R}eg$ is a notational variant of regular grammar logics [3, 8, 9, 15] and relates to propositional dynamic logic (PDL) [13, 36, 17] and the description logic $\mathcal{ALC}_{reg}$ [6] (which uses regular expressions for role constructions). The computational methods for regular grammar logics [3, 8, 9, 15] can easily be extended to deal with global assumptions for automated reasoning in $\mathcal{R}eg$. The logic $\mathcal{R}eg$ differs from $\mathcal{ALC}_{reg}$ in the same way as regular grammar logics differ from PDL. In general, dealing with $\exists R.C$ is easier when R is a role name (the case of $\mathcal{R}eg$) than when R is a role construction with the star operator (the case of $\mathcal{ALC}_{reg}$). On the other hand, role expressions in $\mathcal{ALC}_{reg}$ are "self-contained", while roles in $\mathcal{R}eg$ depend on an "external" RBox.

In [19], Horrocks and Sattler introduced "acyclic generalized RBoxes", and in [18], Horrocks et al. introduced an extended notion called "acyclic regular RBoxes". It can be shown for the case without inverse roles that both the notions of "acyclic generalized RBox" and "acyclic regular RBoxes" are strictly less general than the notion of "regular RBox". The notion of "regular RBox" used by Krötzsch et al. in [24] is the same as the notion of "acyclic generalized RBox" and hence different from our notion of "regular RBox".

$(\mathcal{R}, \mathcal{T}', \mathcal{A})$: a Horn knowledge base in $\mathcal{R}eg$
$\mathcal{T}$: the deterministic version of $\mathcal{T}'$
$\mathcal{I}_1$: a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$
$\mathcal{I}_2$: a least $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$
C : a positive formula (possibly with $\forall$ and $\forall\exists$)
C' : the formula obtained from C by replacing every subformula of the form $\forall\exists R_t.D$ by $(\forall R_t.D \sqcap \exists R_t.D)$

By Lemma 6.10:

$$\begin{array}{ccc} (\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C'(a) & & \\ \Uparrow & & \\ (\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a) & \Leftarrow & \mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a) \\ \Uparrow & & \Uparrow \\ (\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a) & \Leftarrow & \mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a) \end{array}$$

and when C is a positive allsome-formula (by Theorem 8.1):

$$\begin{array}{ccc} (\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C'(a) & & \\ \Uparrow & & \\ (\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a) & \Leftarrow & \mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a) \\ \Uparrow & & \Uparrow \\ (\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a) & \Leftrightarrow & \mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a) \end{array}$$

By Theorem 8.4, if for every $t \in \mathrm{IND}$,

- either the constructor $\forall R_t.D$ does not occur in C' and the premises of the program clauses of $\mathcal{T}'$
- or $\exists R_s.\top \in \mathcal{T}'$ for every $s \in \mathrm{IND}$ occurring in some word accepted by $\mathbf{A}_t$
- or $\mathcal{L}(\mathbf{A}_t) = \{t\}$ and the constructor $\forall R_t.D$ may occur in C' and the premises of the program clauses of $\mathcal{T}'$ only in the form $(\forall R_t.D \sqcap \exists R_t.D')$ for some arbitrary D',

then

$$\begin{array}{ccc} (\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C'(a) & & \\ \Updownarrow & & \\ (\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a) & \Leftrightarrow & \mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a) \\ \Updownarrow & & \Updownarrow \\ (\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a) & \Leftrightarrow & \mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a) \end{array}$$

Fig. 1. Characterizations of least pseudo-models.

A number of Horn fragments of DLs with PTime data complexity (for instance checking or conjunctive query answering – a problem more general than instance checking) have been studied in [16, 20, 5, 22, 24, 37]. The combined complexities of Horn fragments of DLs were studied, amongst others, in [23]. Some Horn fragments of DLs without ABoxes that have PTime complexity have also been studied in [4, 1]. We give here some more details:

- In [16], Grosof et al. introduced the description Horn logic (DHL), which is a restricted fragment of DL, and studied it through a transformation to classical Horn logic. A DHL program consists of Horn clauses specifying (relations between) concepts, (relations between) roles, and instances of concepts and roles. Inverse roles and transitive roles are allowed in DHL programs. In order to make the transformation possible, the constructor $\forall R.C$ is disallowed in premises and the constructor $\exists R.C$ is disallowed in conclusions of DHL program clauses.
- In [20], Hustadt et al. introduced a Horn fragment of the description logic $\mathcal{SHIQ}$, which is called Horn-$\mathcal{SHIQ}$, and studied it using a transformation to Datalog. In comparison with DHL, Horn-$\mathcal{SHIQ}$ also disallows the constructor $\forall R.C$ in premises of program clauses and goals (by a *goal* we call a positive formula C given for the instance checking problem $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$), but it allows the constructor $\exists R.C$ to appear in conclusions of program clauses. There are some additional restrictions for Horn-$\mathcal{SHIQ}$, for example:
  - The constructor $\forall R.C$ may occur in conclusions of program clauses only when R is a simple role (i.e., R and sub-roles of R cannot be transitive).
  - Despite that number restrictions of the form $\geq n\, R.C$ or $\leq 1\, R$ may occur in conclusions of program clauses, the constructor $\leq n\, R.C$ for any n and the constructor $\geq n\, R.C$ for $n \geq 2$ cannot occur in premises of program clauses. (Note that $\geq 1\, R.C$ is equiv to $\exists R.C$.) Thus, the usefulness of the number restriction constructors in Horn-$\mathcal{SHIQ}$ programs is questionable.
- In [5], Calvanese et al. also studied data complexity of query answering in a family called *DL-Lite* of DLs. To obtain low data complexity they adopted strong restrictions for the form of Horn clauses: for example, both the constructors $\forall R.C$ and $\exists R.C$ are not allowed in the left hand side of $\sqsubseteq$, and the constructor $\forall R.C$ is not allowed in the right hand side of $\sqsubseteq$.
- The $\mathcal{EL}$ family of DLs with PTime data complexity studied in [4, 1, 22, 24, 37] completely disallows the constructor $\forall R.C$ in program clauses.

The logic $\mathcal{R}eg$ considered in this work differs from the DLs with PTime data complexity considered in [16, 20, 5, 24, 37] in that, our notion of regular RBox is more general than the notion of RBox of transitive roles and role hierarchies used in [16, 20, 5, 37] and the notion of acyclic generalized RBox used in [24] (and called there "regular RBox"), but on the other hand, in comparison with DHL and Horn-$\mathcal{SHIQ}$, $\mathcal{R}eg$ does not deal with inverse roles. However, the special feature of this work is that we try to allow the constructor $\forall R.C$ or its stronger form $\forall\exists R.C$ in premises of program clauses and goals, while all of DHL [16], Horn-$\mathcal{SHIQ}$ [20], *DL-Lite* [5] and $\mathcal{EL}$ [22, 24, 37] disallow the constructor $\forall R.C$ (and do not use $\forall\exists R.C$) in premises of program clauses and goals. Besides, this work uses a direct approach, while [16, 20] use the translation approach.

For the connection with classical logic programming, note that the functional translation [7] and the semi-functional translation [35] do not work for the general Horn fragment

of $\mathcal{R}eg$ because $\mathcal{R}eg$ is not "serial" (i.e. neither $\exists R_t.\top$ nor $\forall x \exists y\, R_t(x,y)$ is assumed), and the relational translation when applied to a positive logic program or a deterministic positive logic program in $\mathcal{R}eg$ does not produce a Datalog program (because resulting clauses may contain Skolem function symbols and more than one positive literal due to formulas of the form $\forall R.C$ or $\forall \exists R.C$ in premises of the original program clauses).

The "allsome" constructor $\forall \exists R.C$ w.r.t. Sem_1 was introduced in [4]. That work, however, does not deal with the data complexity of the instance checking problem.

This work is a continuation of our previous works [29, 31, 34] and is an extension of our workshop paper [33] and poster [32]. In [29], we introduced and studied the "deterministic Horn fragment of $\mathcal{ALC}$", and in [31] the "deterministic Horn fragment of test-free PDL".[3] In [34], we studied the problem of constructing finite least Kripke models for positive logic programs in serial regular grammar logics. The semantics Sem_1 for $\mathcal{R}eg$ follows [29], while the semantics $\mathsf{Sem}_{2,\mathcal{R}}$ for $\mathcal{R}eg$ follows [31]. This work differs from [29, 31, 34] in an important aspect that it deals with ABoxes and the data complexity of the instance checking problem. Furthermore, note that:

- The logic considered in [29] is $\mathcal{ALC}$ with $\mathcal{R} = \emptyset$.
- The modal logics studied in [34] are serial, while $\mathcal{R}eg$ is non-serial.
- The works [31, 34] deal with logic programs treated as *local assumptions* (for the actual world) but not as a TBox of *global assumptions* (for all individuals).
- Regular RBoxes are different from regular expressions of test-free PDL used in [31] and make the proofs of this paper more complicated than the ones of [31].

In the recent works [11, 12], together with Dunin-Kęplicz and Szałas we studied the data complexity of the Horn fragment of serial PDL. The method used in [11, 12] is an adaptation of our method to serial PDL. Also note that serial PDL is substantially different from (non-serial) $\mathcal{R}eg$.

5 Examples of Application

5.1 Example: Movability of Objects

In this subsection, we present an example about movability of objects, which is a modified version of the one given in [11]. To reason about similarities between objects as in applications of rough sets we can assume that every object is similar to some object (e.g. to itself).

Consider the following scenario:

> Two robots, r_1 and r_2, have the goal to move objects from one place to another. Each robot is able to move objects of a specific signature,[4] and together they might be able to move objects of a combined signature. Robots are working independently, but sometimes have to cooperate to achieve their goals.
>
> To design such robots one has to make a number of decisions as described below.

[3] In [31] we use the term "serial positive formula" and in [29] we use the term "(modal-)deterministic positive concept" instead of "positive allsome-formula".

[4] For example, dependent on weight, size and type of surface.

We assume that the signature of movable objects for each robot is given by its specification together with a similarity relation defining the range of movable objects. Assume the following specification:

$$spec_1 \stackrel{\text{def}}{=} (light \sqcap smooth) \sqcup (heavy \sqcap rough) \ - \text{ for robot } r_1 \tag{1}$$

$$spec_2 \stackrel{\text{def}}{=} small \sqcup medium \ - \text{ for robot } r_2. \tag{2}$$

Movable objects are then specified by

$$spec_i \sqsubseteq movable_i \tag{3}$$

where $i \in \{1, 2\}$ and $movable_i$ is true for objects that can be moved by r_i.

The idea is that all objects similar to movable ones are movable too.[5] We use R_1 and R_2 as roles representing the similarity relations reflecting perceptual capabilities of robots r_1 and r_2, respectively (for a discussion of such similarity relations based on various sensor models see [10]). That is, a role assertion $R_i(o, o')$ means that object o' is perceived similar to object o by robot r_i. Now, in addition to (3), movable objects are characterized by

$$\exists R_i.spec_i \sqsubseteq movable_i. \tag{4}$$

Observe that in general it is impossible to automatically derive combined signatures that specify what robots can move together. Therefore, we introduce $spec_3$ and R_3 as a specification of such joint capabilities. An example of $spec_3$ can be given by

$$spec_3 \stackrel{\text{def}}{=} large \sqcap rough. \tag{5}$$

We shall assume that

$$R_1 \sqsubseteq R_3 \tag{6}$$

$$R_2 \sqsubseteq R_3 \tag{7}$$

$$R_1 \circ R_2 \sqsubseteq R_3 \tag{8}$$

$$R_2 \circ R_1 \sqsubseteq R_3. \tag{9}$$

Of course, one can give another specification for R_3.

Objects movable by robots working together are then defined by

$$spec_3 \sqcup \exists R_3.spec_3 \sqsubseteq movable\_by\_two. \tag{10}$$

Let $\mathcal{T}$ be the TBox consisting of (3), (4), $\exists R_i.\top$ for $i \in \{1, 2\}$, and (10). Thus, $\mathcal{T}$ is a deterministic positive logic program consisting of $\exists R_1.\top$, $\exists R_2.\top$, and

$$\begin{aligned}
(light \sqcap smooth) \sqcup (heavy \sqcap rough) &\sqsubseteq movable_1 \\
\exists R_1.((light \sqcap smooth) \sqcup (heavy \sqcap rough)) &\sqsubseteq movable_1 \\
small \sqcup medium &\sqsubseteq movable_2 \\
\exists R_2.(small \sqcup medium) &\sqsubseteq movable_2 \\
(large \sqcap rough) \sqcup \exists R_3.(large \sqcap rough) &\sqsubseteq movable_by_two.
\end{aligned}$$

[5] This is a very natural and quite powerful technique, allowing one to express the inheritance of particular properties of objects by similar objects.

Let $\mathcal{R}$ be the RBox consisting of role axioms (6)-(9). Clearly, it is a regular RBox. Note that every model of $\mathcal{R}$ and $\mathcal{T}$ is also a model of $\exists R_3.\top$. Let $\mathcal{A}$ be an arbitrary ABox. Thus $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ is a deterministic Horn knowledge base. According to the declared results, for every positive formula C without $\forall\exists$ and for every individual a, we have that $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$ iff $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$, where $\mathcal{I}_1$ is a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.

5.2 Example: Web Pages

There are domains for which the assumption that $\exists R_t.\top \in \mathcal{T}'$ for all $t \in \mathrm{IND}$ is not very natural but somehow acceptable. For example, despite that in general there are web pages without outgoing links, it is not too restrictive to assume that every web page has an outgoing link (e.g. to itself). In this subsection, we present an example about web pages. It is constructed around the atomic concept *interesting*.

Let $\mathcal{R}$ be the regular RBox consisting of the following role axioms:

$$\begin{array}{l} \mathit{link} \sqsubseteq \mathit{path} \\ \mathit{link} \circ \mathit{path} \sqsubseteq \mathit{path}. \end{array}$$

This RBox "defines" *path* to be the transitive closure of *link*. It is a regular RBox specified by the following finite automata, where $\mathrm{IND} = \{l, p\}$, with R_l standing for *link* and R_p standing for *path*:

$$\begin{array}{l} \mathbf{A}_l = \langle \mathrm{IND}, \{1,2\}, \{1\}, \{(1,l,2)\}, \{2\} \rangle \\ \mathbf{A}_p = \langle \mathrm{IND}, \{1,2\}, \{1\}, \{(1,l,1), (1,l,2), (1,p,2)\}, \{2\} \rangle. \end{array}$$

Let $\mathcal{T}'$ be the TBox consisting of the following program clauses:

$$\mathit{perfect} \sqsubseteq \mathit{interesting} \sqcap \forall \mathit{path}.\mathit{interesting} \tag{11}$$

$$\mathit{interesting} \sqcap \forall \mathit{path}.\mathit{interesting} \sqsubseteq \mathit{perfect} \tag{12}$$

$$\mathit{interesting} \sqcup \forall\, \mathit{link}.\mathit{interesting} \sqsubseteq \mathit{worth\_surfing} \tag{13}$$

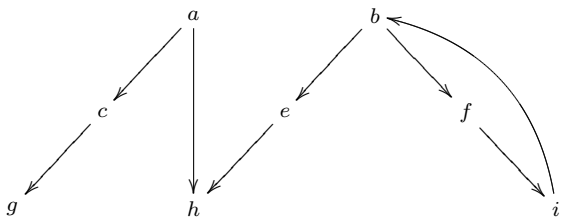
$$\exists \mathit{path}.\mathit{interesting} \sqsubseteq \mathit{has\_a\_path\_to\_an\_interesting\_page} \tag{14}$$

$$\mathit{has\_a\_path\_to\_an\_interesting\_page} \sqsubseteq \exists \mathit{path}.\mathit{interesting}. \tag{15}$$

Note that $\mathcal{T}'$ is equivalent to the TBox consisting of the following:

$$\begin{array}{l} \mathit{perfect} \doteq (\mathit{interesting} \sqcap \forall \mathit{path}.\mathit{interesting}) \\ \mathit{interesting} \sqcup \forall\, \mathit{link}.\mathit{interesting} \sqsubseteq \mathit{worth\_surfing} \\ \mathit{has\_a\_path\_to\_an\_interesting\_page} \doteq \exists \mathit{path}.\mathit{interesting} \end{array}$$

Let $\mathcal{A}$ be the ABox specified by the concept assertions $\mathit{perfect}(b)$, $\mathit{has\_a\_path\_to\_an\_interesting\_page}(g)$ and the following role assertions of *link*:



It can be seen that the used atomic concepts have the following instances w.r.t. the Horn knowledge base $(\mathcal{R}, \mathcal{T}', \mathcal{A})$:

$$perfect, interesting, worth\_surfing : b, e, f, h, i$$
$$has\_a\_path\_to\_an\_interesting\_page : a, b, c, e, f, g, i$$

Note that h is an instance of *perfect* w.r.t. $(\mathcal{R}, \mathcal{T}', \mathcal{A})$ due to the program clause (12), but this clause does not reflect our intention. The deterministic version $\mathcal{T}$ of $\mathcal{T}'$ changes the clauses (12) and (13) of $\mathcal{T}'$ to:

$$interesting \sqcap \forall\exists path.interesting \sqsubseteq perfect$$
$$interesting \sqcup \forall\exists link.interesting \sqsubseteq worth\_surfing$$

With this change, h is no longer an instance of *perfect* w.r.t. the deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, using any semantics $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$.

For this example, approximating the checking problem $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C(a)$ by $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a)$ is justifiable because $\mathcal{T}$ better reflects our intention than $\mathcal{T}'$. But there may still be a gap between $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a)$ and $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$, where $\mathcal{I}_1$ is a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.

Let $\mathcal{T}'_*$ be $\mathcal{T}'$ extended with $\exists link.\top$. Consider the situation when $\mathcal{T}'_*$ is used instead of $\mathcal{T}'$ (i.e. when every web page is assumed to contain at least one link). Let $\mathcal{T}_*$ be the deterministic version of $\mathcal{T}'_*$. Since $link \sqsubseteq path$, every model of $\mathcal{R}$ and $\mathcal{T}'_*$ is also a model of $\exists path.\top$, and we can thus assume that $\exists path.\top \in \mathcal{T}'_*$. Let $\mathcal{I}_*$ be a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}_*, \mathcal{A})$. Then, according to the declared results, for every positive formula C without $\forall\exists$ and for every individual a, $(\mathcal{R}, \mathcal{T}'_*, \mathcal{A}) \models C(a)$ iff $\mathcal{I}_* \models_{\mathsf{Sem}_1} C(a)$. That is, in this case, the approximation is exact.

5.3 Example: Epistemic Reasoning

Regular description logics extended with the seriality axioms $\exists R_t.\top$ for all $t \in \mathrm{IND}$ can be used for epistemic reasoning. For this purpose, individuals and elements of the domain of an interpretation are used to denote possible worlds, and roles are used to denote accessibility relations between possible worlds of agents and groups of agents. Each $t \in \mathrm{IND}$ represents an agent or a group of agents. If t is an agent then a formula $\forall R_t.C$ states that in the current world the agent believes in C (the formal semantics of the formula is that the property C holds for all possible worlds accessible from the current world via the accessibility relation of agent t). If t is a group of agents then the formula $\forall R_t.C$ means that the group "commonly believes" in C.[6] The seriality axiom $\exists R_t.\top$ is equivalent to $\forall R_t.C \sqsubseteq \neg\forall R_t.\neg C$, which states that if t believes in C then it does not believe in $\neg C$. Using $\mathcal{R}eg$ extended with the seriality axioms as an epistemic logic, individuals and ABoxes do not anymore play an important role in complexity issues because usually only the actual world is explicitly used as an individual.

In this subsection, we formalize the wise men puzzle using the general Horn fragment of $\mathcal{R}eg$. This demonstrates the usefulness of the fragment for epistemic reasoning. The puzzle is a famous benchmark of AI introduced by McCarthy [26]. It can be stated as follows (cf. [21]):

[6] Not all properties of common beliefs are expressible in $\mathcal{R}eg$, but we need only some properties of common beliefs for the example considered in this subsection.

A king wishes to know whether his three advisors, denoted by σ_1, σ_2, σ_3, are as wise as they claim to be. Three chairs are lined up, all facing the same direction, with one behind the other. The wise men are instructed to sit down in the order σ_1, σ_2, σ_3 with σ_1 on front. Each of the men can see the backs of the men sitting before them (e.g. σ_3 can see σ_2 and σ_1). The king informs the wise men that he has three cards, all of which are either black or white, at least one of which is white. He places one card, face up, behind each of the three wise men, explaining that each wise man must determine the color of his own card. Each wise man must announce the color of his own card as soon as he knows what it is. All know that this will happen. The room is silent; then, after a while, wise man σ_1 says "My card is white!".

We use $\mathrm{IND} = \{1, 2, 3, g\}$. For $t \in \{1, 2, 3\}$, let $\forall R_t.C$ stand for "the wise man σ_t believes in C", $white_t$ stand for "the card of σ_t is white", and $black_t$ stand for "the card of σ_t is black". Let g denote the group $\{\sigma_1, \sigma_2, \sigma_3\}$ and let $\forall R_g.C$ state that C is a "common belief" of the group g. Let $\mathcal{R}$ be the RBox consisting of the following role axioms:

$$R_t \sqsubseteq R_g \text{ – for } t \in \{1, 2, 3\}$$
$$R_t \circ R_t \sqsubseteq R_t \text{ – for } t \in \{1, 2, 3, g\}.$$

This RBox is regular and specified by the following finite automata:

$$\mathbf{A}_t = \langle \mathrm{IND}, \{1, 2\}, \{1\}, \{(1, t, 1), (1, t, 2)\}, \{2\}\rangle \text{ for } t \in \{1, 2, 3\}$$
$$\mathbf{A}_g = \langle \mathrm{IND}, \{1, 2\}, \{1\}, \{(1, t, 1), (1, t, 2) \mid t \in \{1, 2, 3, g\}\}, \{2\}\rangle.$$

The wise men puzzle can be formalized as follows (cf. [30, 34]).

It is a common belief of the group that if y sits behind x then x's card is white whenever y considers this possible:

$$\forall R_g.(\exists R_2.white_1 \sqsubseteq white_1) \tag{16}$$
$$\forall R_g.(\exists R_3.white_1 \sqsubseteq white_1) \tag{17}$$
$$\forall R_g.(\exists R_3.white_2 \sqsubseteq white_2) \tag{18}$$

The following clauses are "dual" to the above ones:

$$\forall R_g.(black_1 \sqsubseteq \forall R_2.black_1) \tag{19}$$
$$\forall R_g.(black_1 \sqsubseteq \forall R_3.black_1) \tag{20}$$
$$\forall R_g.(black_2 \sqsubseteq \forall R_3.black_2) \tag{21}$$

It is a common belief of the group that at least one of the wise men has a white card:

$$\forall R_g.(black_2, black_3 \sqsubseteq white_1) \tag{22}$$
$$\forall R_g.(black_3, black_1 \sqsubseteq white_2) \tag{23}$$
$$\forall R_g.(black_1, black_2 \sqsubseteq white_3) \tag{24}$$

It is a common belief of the group that: each of σ_2 and σ_3 does not know the color of his own card; in particular, each of the men considers that it is possible that his own

card is black:

$$\forall R_g.\exists R_2.black_2 \tag{25}$$
$$\forall R_g.\exists R_3.black_3 \tag{26}$$

The formulas (16)-(24) are supposed to hold for every possible world, while the formulas (25) and (26) are only supposed to hold for the actual world. Since only extensionally reduced ABoxes are allowed, we encode the conjunction of (25) and (26) by an atomic concept A, and assume that our ABox $\mathcal{A}$ is $\{A(\tau)\}$, where τ is the only individual, which represents the actual world, and we treat the following formula as a global assumption:

$$A \sqsubseteq \forall R_g.\exists R_2.black_2 \sqcap \forall R_g.\exists R_3.black_3 \tag{27}$$

Let $\mathcal{T}$ be the deterministic positive logic program consisting of the formulas (16)-(24), (27), and $\exists R_t.\top$ for $t \in \mathrm{IND}$. Thus $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ is a deterministic Horn knowledge base.

The goal is to check whether wise man σ_1 believes that his card is white: that is, whether $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models (\forall R_1.white_1)(\tau)$. According to the declared results, this question is equivalent to checking whether $\mathcal{I}_1 \models_{\mathsf{Sem}_1} (\forall R_1.white_1)(\tau)$, where $\mathcal{I}_1$ is a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. It can be shown that the answer is affirmative.

5.4 Example: Formalizing a Search Problem

In this subsection, we formalize the missionaries and cannibals problem using the general Horn fragment of $\mathcal{R}eg$. The problem is well-known for courses of AI [38] and is stated as follows:

> Three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries.) The boat cannot cross the river by itself with no people on board.

For the considered problem, individuals and elements of the domain of an interpretation are used to denote states of the search space, and roles are used to denote actions. We represent each state by a tuple (M_i, C_j, B_k), where

- $i \in \{0,1,2,3\}$, $j \in \{0,1,2,3\}$, $k \in \{1,2\}$,
- M_i means that there are i missionaries on the departure bank,
- C_j means that there are j cannibals on the departure bank,
- B_1 means that the boat is at the departure bank,
- B_2 means that the boat is at the arrival bank.

We use the following roles:

- R_{m_i,c_j} : the boat takes m_i missionaries and c_j cannibals to the other bank $(1 \leq i + j \leq 2)$,
- R_{m_i} : the boat takes m_i missionaries (and possibly some cannibals) to the other bank $(0 \leq i \leq 2)$,

- R_{c_i} : the boat takes c_i cannibals (and possibly some missionaries) to the other bank ($0 \leq i \leq 2$),
- R_a : the boat moves to the other bank (the subscript a stands for "action"),
- R_+ : a nonempty sequence of moves of the boat between the banks.

Let $\mathcal{R}$ be the RBox consisting of the following role axioms:

$$\begin{aligned} R_{m_i,c_j} &\sqsubseteq R_{m_i} \\ R_{m_i,c_j} &\sqsubseteq R_{c_j} \\ R_{m_i} &\sqsubseteq R_a \\ R_{c_i} &\sqsubseteq R_a \\ R_a &\sqsubseteq R_+ \\ R_a \circ R_+ &\sqsubseteq R_+ \end{aligned}$$

It can be seen that $\mathcal{R}$ is a regular RBox.

Let $\mathcal{T}$ be the deterministic positive logic program consisting of the following clauses:

$$B_1 \sqsubseteq \forall R_a.B_2$$

$$B_2 \sqsubseteq \forall R_a.B_1$$

$$B_1 \sqcap M_i \sqsubseteq \forall R_{m_j}.M_k \text{ for } i \in \{0,1,2,3\}, j \in \{0,1,2\}, i \geq j, k = i - j$$

$$B_2 \sqcap M_i \sqsubseteq \forall R_{m_j}.M_k \text{ for } i \in \{0,1,2,3\}, j \in \{0,1,2\}, 3 - i \geq j, k = i + j$$

$$B_1 \sqcap C_i \sqsubseteq \forall R_{c_j}.C_k \text{ for } i \in \{0,1,2,3\}, j \in \{0,1,2\}, i \geq j, k = i - j$$

$$B_2 \sqcap C_i \sqsubseteq \forall R_{c_j}.C_k \text{ for } i \in \{0,1,2,3\}, j \in \{0,1,2\}, 3 - i \geq j, k = i + j$$

$$B_1 \sqcap M_i \sqcap C_j \sqsubseteq \exists R_{m_{i'},c_{j'}}.\top \text{ for } \begin{cases} i \in \{0,1,2,3\}, i' \in \{0,1,2\}, i \geq i', \\ j \in \{0,1,2,3\}, j' \in \{0,1,2\}, j \geq j', \\ (i - i' = 0) \vee (i - i' \geq j - j'), \\ (3 - i + i' = 0) \vee (3 - i + i' \geq 3 - j + j') \end{cases}$$

$$B_2 \sqcap M_i \sqcap C_j \sqsubseteq \exists R_{m_{i'},c_{j'}}.\top \text{ for } \begin{cases} i \in \{0,1,2,3\}, i' \in \{0,1,2\}, 3 - i \geq i', \\ j \in \{0,1,2,3\}, j' \in \{0,1,2\}, 3 - j \geq j', \\ (3 - i - i' = 0) \vee (3 - i - i' \geq 3 - j - j'), \\ (i + i' = 0) \vee (i + i' \geq j + j') \end{cases}$$

Let s be an individual representing the initial state and let

$$\mathcal{A} = \{B_1(s), M_3(s), C_3(s)\}$$

Thus $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ is a deterministic Horn knowledge base. The question is whether there exists a solution for the problem, which is formalized as the problem of checking whether

$$(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models (\exists R_+.(B_2 \sqcap M_0 \sqcap C_0))(s)$$

According to the declared results, this problem is equivalent to checking whether

$$\mathcal{I}_1 \models_{\mathsf{Sem}_1} (\exists R_+.(B_2 \sqcap M_0 \sqcap C_0))(s)$$

where $\mathcal{I}_1$ is a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. It can be shown that the answer is affirmative.

Similarly as for epistemic reasoning, for formalizing search problems individuals and ABoxes do not anymore play an important role in complexity issues because usually only the initial state is explicitly used as an individual. The example, however, demonstrates the expressiveness of the general Horn fragment of $\mathcal{R}eg$.

5.5 Example: Constructivism

Let $\mathcal{T}'$ be the positive logic program consisting of the following clauses:

$$\exists \mathit{has\_child}.\top \sqsubseteq \mathit{parent} \tag{28}$$

$$\forall \mathit{has\_child}.(\mathit{doctor} \sqcup \mathit{lawyer}) \sqsubseteq \mathit{happy\_parent} \tag{29}$$

$$\mathit{all\_children\_are\_lawyers} \sqsubseteq \forall \mathit{has\_child}.\mathit{lawyer} \tag{30}$$

Let $\mathcal{R} = \emptyset$ and

$$\mathcal{A} = \{\mathit{has\_child}(\mathit{Jane}, \mathit{Peter}), \mathit{has\_child}(\mathit{Jane}, \mathit{Christ}), \\ \mathit{all\_children\_are\_lawyers}(\mathit{Jane})\}$$

Consider the Horn knowledge base $(\mathcal{R}, \mathcal{T}', \mathcal{A})$. As expected, we have that

$$(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models \mathit{happy\_parent}(\mathit{Jane})$$

But unexpectedly, we also have that

$$(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models (\mathit{parent} \sqcup \mathit{happy\_parent})(\mathit{Peter})$$
$$(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models (\mathit{parent} \sqcup \mathit{happy\_parent})(\mathit{Christ})$$

The reason is that the premise of the program clause (29) is not "constructive". In other words, the instance $\exists \mathit{has\_child}.\top \sqcup \forall \mathit{has\_child}.\neg\top$ of the "law of excluded middle" holds for every individual a. If a satisfies $\exists \mathit{has\_child}.\top$ then it is an instance of *parent*; else it satisfies $\forall \mathit{has\_child}.\neg\top$, and therefore satisfies any formula $\forall \mathit{has\_child}.C$, and hence is an instance of *happy_parent* (by (29)).

The program clause (29) should be changed to

$$\mathit{parent} \sqcap \forall \mathit{has\_child}.(\mathit{doctor} \sqcup \mathit{lawyer}) \sqsubseteq \mathit{happy\_parent}$$

or equivalently w.r.t. Sem_1, given (28), to

$$\forall\exists \mathit{has\_child}.(\mathit{doctor} \sqcup \mathit{lawyer}) \sqsubseteq \mathit{happy\_parent} \tag{31}$$

which is the deterministic version of (29).

The deterministic version of $\mathcal{T}'$ is $\mathcal{T}$ consisting of the program clauses (28), (30), (31). According to the declared results (the third case of the last diagram of Figure 1), for any positive formula C possibly with the constructor $\forall\exists$ but without $\forall$ and for any individual a, $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a)$ iff $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$, where $\mathcal{I}_1$ is a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.

The above example shows that, in premises of program clauses and goals (the positive formulas given for instance checking), the constructor $\forall\exists$ is more "constructive" than $\forall$.

6 Pseudo-Interpretations

Definition 6.1. A *pseudo-interpretation* is a tuple $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$, where Δ is a non-empty set representing the domain, $\mathcal{O}$ maps every individual to an element of Δ, $\mathcal{C}$ maps every atomic concept to a subset of Δ, and $\mathcal{E}$ and $\mathcal{U}$ map every role name to a subset of $\Delta \times \Delta$, with the property that $\mathcal{E}(R_t) \subseteq \mathcal{U}(R_t)$ for every role name R_t. By $a^{\mathcal{I}}$ we denote $\mathcal{O}(a)$. ◁

The function $\mathcal{E}$ is used to deal with the (**e**xistential) constructor $\exists$, while $\mathcal{U}$ is used to deal with the (**u**niversal) constructor $\forall$. The intuition behind the mappings $\mathcal{E}$ and $\mathcal{U}$ is as follows: every edge created to satisfy a formula of the form $\exists R_t.C$ is included into $\mathcal{E}(R_t)$, but we also want to use additional edges for R_t; the set $\mathcal{U}(R_t)$ is the set $\mathcal{E}(R_t)$ plus the additional edges created for R_t, which causes $\mathcal{E}(R_t) \subseteq \mathcal{U}(R_t)$.

A pseudo-interpretation $\langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ can be treated as an interpretation if $\mathcal{E} = \mathcal{U}$. Conversely, every interpretation can be treated as a pseudo-interpretation.

Definition 6.2. A pseudo-interpretation $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ is a *pseudo-model of an RBox* $\mathcal{R}$ iff $\mathcal{E}(R_{s_1}) \circ \ldots \circ \mathcal{E}(R_{s_k}) \subseteq \mathcal{E}(R_t)$ and $\mathcal{U}(R_{s_1}) \circ \ldots \circ \mathcal{U}(R_{s_k}) \subseteq \mathcal{U}(R_t)$ for every role axiom $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ of $\mathcal{R}$. It is a *pseudo-model of an ABox* $\mathcal{A}$ iff $a^{\mathcal{I}} \in \mathcal{C}(A)$ for every $A(a) \in \mathcal{A}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \mathcal{E}(R)$ for every $R(a,b) \in \mathcal{A}$. ◁

Given a pseudo-model $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ of an RBox $\mathcal{R}$, an element $x \in \Delta$, and a formula C which is either a positive formula (possibly with $\forall$ and $\forall\exists$) or a deterministic program clause, define $\mathcal{I}, x \models_{\mathfrak{s}} C$ as follows:

$$
\begin{array}{lll}
\mathcal{I}, x \models_{\mathfrak{s}} A & \text{iff} & x \in \mathcal{C}(A) \\
\mathcal{I}, x \models_{\mathfrak{s}} C \sqcap D & \text{iff} & \mathcal{I}, x \models_{\mathfrak{s}} C \text{ and } \mathcal{I}, x \models_{\mathfrak{s}} D \\
\mathcal{I}, x \models_{\mathfrak{s}} C \sqcup D & \text{iff} & \mathcal{I}, x \models_{\mathfrak{s}} C \text{ or } \mathcal{I}, x \models_{\mathfrak{s}} D \\
\mathcal{I}, x \models_{\mathfrak{s}} C \sqsubseteq D & \text{iff} & \mathcal{I}, x \not\models_{\mathfrak{s}} C \text{ or } \mathcal{I}, x \models_{\mathfrak{s}} D \\
\mathcal{I}, x \models_{\mathfrak{s}} \exists R_t.C & \text{iff} & \exists y.(\mathcal{E}(R_t)(x,y) \wedge \mathcal{I}, y \models_{\mathfrak{s}} C) \\
\mathcal{I}, x \models_{\mathfrak{s}} \forall R_t.C & \text{iff} & \forall y.(\mathcal{U}(R_t)(x,y) \rightarrow \mathcal{I}, y \models_{\mathfrak{s}} C) \\
\mathcal{I}, x \models_{\mathfrak{s}} \forall\exists R_t.C & \text{iff} & \mathcal{I}, x \models_{\mathfrak{s}} D \text{ for every } D \in \mathfrak{s}(\forall\exists R_t.C)
\end{array}
$$

For the notion of $\mathfrak{s}(\_)$, see Definition 3.1.
We write $\mathcal{I} \models_{\mathfrak{s}} C(a)$ for $\mathcal{I}, a^{\mathcal{I}} \models_{\mathfrak{s}} C$.

Definition 6.3. We say that a pseudo-interpretation $\mathcal{I}$ with domain Δ *validates* a formula C w.r.t. $\mathfrak{s}$ if $\mathcal{I}, x \models_{\mathfrak{s}} C$ for every $x \in \Delta$. For a deterministic positive logic program $\mathcal{T}$, we say that $\mathcal{I}$ is an $\mathfrak{s}$-*pseudo-model of* $\mathcal{T}$ if $\mathcal{I}$ validates all formulas of $\mathcal{T}$ w.r.t. $\mathfrak{s}$. ◁

Definition 6.4. A pseudo-interpretation is a $\mathfrak{s}$-*pseudo-model* of a deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ if it is a pseudo-model of $\mathcal{R}$ and $\mathcal{A}$ and is an $\mathfrak{s}$-pseudo-model of $\mathcal{T}$. ◁

6.1 Ordering Pseudo-Interpretations

Definition 6.5. Let $\mathcal{I}$ and $\mathcal{I}'$ be pseudo-models of an RBox $\mathcal{R}$. We say that $\mathcal{I}$ is *less than or equal to* $\mathcal{I}'$ w.r.t. semantics $\mathfrak{s}$, write $\mathcal{I} \leq_{\mathfrak{s}} \mathcal{I}'$, if for every positive formula C (possibly with $\forall$ and $\forall\exists$) and every individual a, $\mathcal{I} \models_{\mathfrak{s}} C(a)$ implies $\mathcal{I}' \models_{\mathfrak{s}} C(a)$.[7] ◁

[7] In [29], we use a weaker ordering in the form "a pseudo-interpretation $\mathcal{I}$ is less than or equal to a pseudo-interpretation $\mathcal{I}'$ if for every positive formula C, if $\mathcal{I}$ validates C then $\mathcal{I}'$ also validates C".

Definition 6.6. A pseudo-interpretation $\mathcal{I}$ is called a *least $\mathfrak{s}$-pseudo-model* of a deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ if it is an $\mathfrak{s}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ and for every $\mathfrak{s}$-pseudo-model $\mathcal{I}'$ of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, $\mathcal{I} \leq_{\mathfrak{s}} \mathcal{I}'$. ◁

Note that $\mathcal{I}$ and $\mathcal{I}'$ are least $\mathfrak{s}$-pseudo-models of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ does not imply $\mathcal{I} = \mathcal{I}'$, as it only states that, for every positive formula C and every individual a, $\mathcal{I} \models_{\mathfrak{s}} C(a)$ iff $\mathcal{I}' \models_{\mathfrak{s}} C(a)$.

Definition 6.7. Let $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ and $\mathcal{I}' = \langle \Delta', \mathcal{O}', \mathcal{C}', \mathcal{E}', \mathcal{U}' \rangle$ be pseudo-interpretations. We say that $\mathcal{I}$ is *less than or equal to $\mathcal{I}'$ w.r.t. a binary relation* $r \subseteq \Delta \times \Delta'$, and write $\mathcal{I} \leq_r \mathcal{I}'$, if the following conditions hold for every individual a, every role name R_t and every atomic concept A:

1. $r(a^{\mathcal{I}}, a^{\mathcal{I}'})$
2. $\forall x, x', y \;\; \mathcal{E}(R_t)(x, y) \wedge r(x, x') \rightarrow \exists y' \; \mathcal{E}'(R_t)(x', y') \wedge r(y, y')$
3. $\forall x, x', y' \;\; \mathcal{U}'(R_t)(x', y') \wedge r(x, x') \rightarrow \exists y \, \mathcal{U}(R_t)(x, y) \wedge r(y, y')$
4. $\forall x, x' \;\; r(x, x') \rightarrow (x \in \mathcal{C}(A) \rightarrow x' \in \mathcal{C}'(A))$ ◁

In the above definition, the first three conditions state that r is a kind of bisimulation (forward w.r.t. $\mathcal{E}/\mathcal{E}'$ and backward w.r.t. $\mathcal{U}/\mathcal{U}'$) of the "frames" of $\mathcal{I}$ and $\mathcal{I}'$. Intuitively, $r(x, x')$ states that if x is an instance of a positive formula C (understood as a "concept") in $\mathcal{I}$ then x' is an instance of C in $\mathcal{I}'$.

Lemma 6.8. *Let $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ and $\mathcal{I}' = \langle \Delta', \mathcal{O}', \mathcal{C}', \mathcal{E}', \mathcal{U}' \rangle$ be pseudo-models of an RBox $\mathcal{R}$. Suppose that $\mathcal{I} \leq_r \mathcal{I}'$ for some r. Then $\mathcal{I} \leq_{\mathfrak{s}} \mathcal{I}'$ (for both $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$). Moreover, for every $x \in \Delta$ and $x' \in \Delta'$, if $r(x, x')$ holds then, for every positive formula C (possibly with $\forall$ and $\forall\exists$), $\mathcal{I}, x \models_{\mathfrak{s}} C$ implies $\mathcal{I}', x' \models_{\mathfrak{s}} C$.*

Proof. Let C be an arbitrary positive formula and suppose that $r(x, x')$ holds. We prove that $\mathcal{I}, x \models_{\mathfrak{s}} C$ implies $\mathcal{I}', x' \models_{\mathfrak{s}} C$ by induction on the construction of C. Suppose that $\mathcal{I}, x \models_{\mathfrak{s}} C$.

The cases when C is of the form A, $C' \sqcap C''$, or $C' \sqcup C''$ are trivial.

Case $C = \exists R_t.D$: Since $\mathcal{I}, x \models_{\mathfrak{s}} C$, there exists $y \in \Delta$ such that $\mathcal{E}(R_t)(x, y)$ holds and $\mathcal{I}, y \models_{\mathfrak{s}} D$. By Condition 2 of the definition of $\leq_r$, there exists $y' \in \Delta'$ such that $\mathcal{E}'(R_t)(x', y')$ and $r(y, y')$ hold. By the inductive assumption, $\mathcal{I}', y' \models_{\mathfrak{s}} D$, hence $\mathcal{I}', x' \models_{\mathfrak{s}} C$.

Case $C = \forall R_t.D$: Let y' be an arbitrary element of Δ' such that $\mathcal{U}'(R_t)(x', y')$ holds. By Condition 3 of the definition of $\leq_r$, there exists $y \in \Delta$ such that $\mathcal{U}(R_t)(x, y)$ and $r(y, y')$ hold. Thus, $\mathcal{I}, y \models_{\mathfrak{s}} D$, and by the inductive assumption, $\mathcal{I}', y' \models_{\mathfrak{s}} D$, hence $\mathcal{I}', x' \models_{\mathfrak{s}} C$.

The case when C is of the form $\forall\exists R_t.D$ is reduced to the two above cases. ◁

6.2 Some Properties

Mappings and relations are treated as sets. If not stated otherwise, the *size of a set* X is the number of elements of X, denoted by $|X|$. The *length of a formula* is the number of symbols occurring in the formula. The *size of a pseudo-interpretation* $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ is defined to be

$$|\Delta| + |\mathcal{O}| + \Sigma_{\mathcal{C}(A) \neq \emptyset} |\mathcal{C}(A)| + \Sigma_{t \in \mathrm{IND}} |\mathcal{E}(R_t)| + \Sigma_{t \in \mathrm{IND}} |\mathcal{U}(R_t)|.$$

Proposition 6.9. *Let $\mathcal{I}$ be a pseudo-model of a regular RBox $\mathcal{R}$, C a positive formula (possibly with $\forall$ and $\forall\exists$), a an individual, and $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$. Then checking whether $\mathcal{I} \models_{\mathfrak{s}} C(a)$ can be done in polynomial time in the size of $\mathcal{I}$ and C, assuming that the finite automata specifying $\mathcal{R}$ are fixed.*

Proof sketch. Let $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$. Checking whether $\mathcal{I} \models_{\mathfrak{s}} C(a)$ can be done by computing the set $D^{\mathcal{I}} = \{x \in \Delta \mid \mathcal{I}, x \models_{\mathfrak{s}} D\}$ for each subformula D of C as in dynamic programming. It can be proved by induction on the construction of D that the set $D^{\mathcal{I}}$ can be computed in $O(n^3 \times m)$ steps, where n is the size of Δ and m is the length of D. Here, we consider only the representative case when $D = \forall\exists R_t.D'$ and $\mathfrak{s} = \mathsf{Sem}_{2,\mathcal{R}}$.

Let $\mathbf{A}_t = \langle \mathrm{IND}, Q_t, I_t, \delta_t, F_t \rangle$ be the automaton corresponding to t amongst the ones specifying $\mathcal{R}$. Let $x \in \Delta$ and consider the problem of checking $\mathcal{I}, x \models_{\mathfrak{s}} \forall\exists R_t.D'$. By starting from x and "running" $\mathbf{A}_t$ across edges of $\mathcal{I}$, we can compute in $O(|\Delta| + |\mathcal{U}|) = O(n^2)$ steps the set S_x of all pairs $(y, q) \in \Delta \times Q_t$ such that there is a path in $\mathcal{I}$ from x to y via $\mathcal{U}(R_{s_1}), \ldots, \mathcal{U}(R_{s_i})$ and there is a run of $\mathbf{A}_t$ on the word $s_1 \ldots s_i$ that ends at q. Then, to check $\mathcal{I}, x \models_{\mathfrak{s}} \forall\exists R_t.D'$, apart from checking $\mathcal{I}, x \models_{\mathfrak{s}} \forall R_t.D'$, it suffices to check that $\mathcal{I}, y \models_{\mathfrak{s}} \exists R_s.\top$ for every $(y, q) \in S_x$ and every $(q, s, q') \in \delta_t$ such that there is an accepting run of $\mathbf{A}_t$ starting from q'. As $\mathbf{A}_t$ and IND are fixed, the cost of computing $(\forall\exists R_t.D')^{\mathcal{I}}$ can be estimated as the cost of computing $(\forall R_t.D')^{\mathcal{I}}$ plus $O(n^3)$ for computing the sets S_x for $x \in \Delta$. Without loss of generality, we can assume that $\forall R_t.D'$ is a subformula of $\forall\exists R_t.D'$ with length $m - 1$. By the inductive assumption, $(\forall R_t.D')^{\mathcal{I}}$ can be computed in $O(n^3 \times (m-1))$ steps. Hence $(\forall\exists R_t.D')^{\mathcal{I}}$ can be computed in $O(n^3 \times m)$ steps. ◁

Lemma 6.10. *Let $(\mathcal{R}, \mathcal{T}', \mathcal{A})$ be a Horn knowledge base in $\mathcal{R}eg$, $\mathcal{T}$ the deterministic version of $\mathcal{T}'$, $\mathcal{I}_1$ (resp. $\mathcal{I}_2$) a least Sem_1-pseudo-model (resp. $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model) of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, C a positive formula (possibly with $\forall$ and $\forall\exists$), C' the formula obtained from C by changing every subformula of the form $\forall\exists R_t.D$ to $(\forall R_t.D \sqcap \exists R_t.D)$, and a an individual. Then:*

1. *Every model of $(\mathcal{R}, \mathcal{T}', \mathcal{A})$ is a Sem_1-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, and every Sem_1-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ is a $\mathsf{Sem}_{2,\mathcal{R}}$-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.*
2. *$(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a)$ implies $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C'(a)$, and $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$ implies $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a)$.*
3. *$\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$ implies $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_1} C(a)$, and $\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$ implies $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$.*
4. *$\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$ implies $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$.*

See the first diagram of Figure 1 for an illustration of this lemma.

Proof. Let $\mathcal{I}$ be a model of $\mathcal{R}$. It is easy to prove by induction on the construction of C that, for $x \in \Delta^{\mathcal{I}}$,

$$\text{if } \mathcal{I}, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C \text{ then } \mathcal{I}, x \models_{\mathsf{Sem}_1} C, \tag{32}$$

$$\text{if } \mathcal{I}, x \models_{\mathsf{Sem}_1} C \text{ then } \mathcal{I}, x \models C'. \tag{33}$$

Let D' be a program clause and D the deterministic version of D'. Using (32) and (33), it is easy to prove by induction on the construction of D that, for $x \in \Delta^{\mathcal{I}}$,

$$\text{if } \mathcal{I}, x \models_{\mathsf{Sem}_1} D \text{ then } \mathcal{I}, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} D, \tag{34}$$

$$\text{if } \mathcal{I}, x \models D' \text{ then } \mathcal{I}, x \models_{\mathsf{Sem}_1} D. \tag{35}$$

The first assertion of the lemma immediately follows from (34) and (35).

Consider the second assertion and suppose that $(\mathcal{R},\mathcal{T},\mathcal{A}) \models_{\mathsf{Sem}_1} C(a)$. We show that $(\mathcal{R},\mathcal{T}',\mathcal{A}) \models C'(a)$. Let $\mathcal{I}$ be a model of $(\mathcal{R},\mathcal{T}',\mathcal{A})$. By (35), $\mathcal{I}$ is also a Sem_1-model of $(\mathcal{R},\mathcal{T},\mathcal{A})$. Hence $\mathcal{I} \models_{\mathsf{Sem}_1} C(a)$. By (33), we derive that $\mathcal{I} \models C'(a)$. Hence $(\mathcal{R},\mathcal{T}',\mathcal{A}) \models C'(a)$.

The remaining part of the second assertion can be proved analogously.

For the third assertion, just note that $\mathcal{I}_1$ is a Sem_1-model of $(\mathcal{R},\mathcal{T},\mathcal{A})$ and $\mathcal{I}_2$ is a $\mathsf{Sem}_{2,\mathcal{R}}$-model of $(\mathcal{R},\mathcal{T},\mathcal{A})$.

Consider the fourth assertion and suppose that $\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$. Thus, $(\mathcal{R},\mathcal{T},\mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$, since $\mathcal{I}_2$ is a least $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R},\mathcal{T},\mathcal{A})$. Similarly to the proof of the first assertion, it can be shown that $\mathcal{I}_1$ is a $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R},\mathcal{T},\mathcal{A})$. It follows that $\mathcal{I}_1 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$, and hence $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$. ◁

7 Constructing Finite Least Pseudo-Models

If $\mathbf{A}$ is a finite automaton and Q is a subset of the states of $\mathbf{A}$ then we call $[\mathbf{A},Q]$ a (universal) *automaton-modal operator/constructor*. If C is a formula without automaton-modal operators then we call $[\mathbf{A},Q]C$ a formula (in the extended language). In this section, by a *formula* we mean a formula in the language extended with automaton-modal operators and the constructor $\forall\exists$.

The technique of using automaton-modal formulas for building finite models in modal/description logics that are related with regular languages has been previously used in [17, 19, 15, 18, 31]. It guarantees the subformula property (also called the super-formula property [14]): the set of formulas used to build a model of a finite set Γ of formulas consists of subformulas from a certain finite set (depending on Γ).

Fix a regular RBox $\mathcal{R}$ and let $(\mathbf{A}_t = \langle \mathrm{IND}, Q_t, I_t, \delta_t, F_t\rangle)_{t\in\mathrm{IND}}$ be the automata specifying $\mathcal{R}$. Let $\delta_t(Q,s) = \{q' \mid \exists q \in Q.(q,s,q') \in \delta_t\}$ be the set of all states which can be reached from Q via an s-transition of $\mathbf{A}_t$. Let ε be the empty word and define $\widetilde{\delta}_t(Q,\varepsilon) = Q$ and $\widetilde{\delta}_t(Q, s_1 \dots s_k) = \delta_t(\widetilde{\delta}_t(Q, s_1 \dots s_{k-1}), s_k)$. The formal semantics of formulas with automaton-modal operators are defined as follows.

Definition 7.1. Let $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U}\rangle$ be a pseudo-model of $\mathcal{R}$, and $x_0 \in \Delta$. Define that $\mathcal{I}, x_0 \models_{\mathfrak{s}} [\mathbf{A}_t, Q]C$ if $\mathcal{I}, x_k \models_{\mathfrak{s}} C$ for every path $x_0\mathcal{U}(R_{s_1})x_1 \dots x_{k-1}\mathcal{U}(R_{s_k})x_k$ with $k \geq 0$ such that $\widetilde{\delta}_t(Q, s_1 \dots s_k) \cap F_t \neq \emptyset$ (i.e. $s_1 \dots s_k$ is accepted by $\mathbf{A}_t$ when starting from some state from Q). ◁

In this section, we present an algorithm that constructs a finite least $\mathfrak{s}$-pseudo-model for a given deterministic Horn knowledge base $(\mathcal{R},\mathcal{T},\mathcal{A})$ in $\mathcal{R}eg$.[8] In the algorithm we use the following data structures:

- Δ is a set forming the domain of the constructed pseudo-interpretation. We distinguish a subset Δ_0 of Δ that consists of all the individuals occurring in the ABox $\mathcal{A}$. In the case $\mathcal{A}$ is empty, let $\Delta_0 = \{\tau\}$ for some element τ.
- $\mathbf{C}$ is a mapping that maps every $x \in \Delta$ to a set of formulas. We treat elements of Δ as possible worlds as in modal logic, and $\mathbf{C}(x)$ thus denotes the "content" of the possible world x.

[8] Recall that a deterministic Horn knowledge base may have more than one least $\mathfrak{s}$-pseudo-models.

- $\mathbf{E}$ is a mapping such that for $x \in \Delta$ and $\exists R_t.C \in \mathbf{C}(x)$, $\mathbf{E}(x, \exists R_t.C) \in \Delta$. The meaning of $\mathbf{E}(x, \exists R_t.C) = y$ is that $\exists R_t.C \in \mathbf{C}(x)$, $C \in \mathbf{C}(y)$, and the "requirement" $\exists R_t.C$ is satisfied at x by going to y via R_t (treating x and y as possible worlds).
- $\mathbf{U}$ is a mapping such that for $x \in \Delta$ and a role name R_t, $\mathbf{U}(x, R_t) \in \Delta$. Let us give the intuition behind the use of this mapping. If the content of x contains only $\exists R_t.C$, then by connecting x to y with $C \in \mathbf{C}(y)$, $\exists R_t.C$ will be satisfied at x, but $\forall R_t.C$ may (still) be satisfied at x because at that time the created edge may be the only edge of R_t going out from x, which is undesirable (because $\forall R_t.C$ is not a "logical consequence" of $\exists R_t.C$). The solution is that for every $x \in \Delta$ and every role name R_t, we connect x via R_t to some y with a minimal content forced by the content of x. However, this has the undesirable side effect that $\exists R_t.\top$ is satisfied at x (note that $\exists R_t.\top$ is not assumed to be an "axiom"). Hence we need to distinguish the edge $R_t(x, y)$ from the "normal" edges of R_t and that is why we use both the mappings $\mathbf{E}$ and $\mathbf{U}$.

We call the tuple $\langle \Delta, \mathbf{C}, \mathbf{E}, \mathbf{U} \rangle$ a *model graph*.

The algorithm also uses a pseudo-interpretation $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ defined below, which at the end is the output of the algorithm:

- $\mathcal{C}(A) = \{x \mid A \in \mathbf{C}(x)\}$,
- $\mathcal{O}$ is the function that maps every individual $a \in \Delta_0$ to a,
- $\mathcal{E}_0(R_t) = \{(a, b) \mid R_t(a, b) \in \mathcal{A}\} \cup \{(x, y) \mid \mathbf{E}(x, \exists R_t.C) = y \text{ for some } C\}$,
- $\mathcal{U}_0(R_t) = \mathcal{E}_0(R_t) \cup \{(x, y) \mid \mathbf{U}(x, R_t) = y\}$,
- $(\mathcal{E}(R_t))_{t \in \mathrm{IND}}$ is the least extension of $(\mathcal{E}_0(R_t))_{t \in \mathrm{IND}}$ that satisfies $\mathcal{R}$,
- $(\mathcal{U}(R_t))_{t \in \mathrm{IND}}$ is the least extension of $(\mathcal{U}_0(R_t))_{t \in \mathrm{IND}}$ that satisfies $\mathcal{R}$.

Definition 7.2. The *saturation* of a set Γ of formulas, denoted by $\mathsf{Sat}(\Gamma)$, is defined to be the smallest superset of Γ such that:

- $\top \in \mathsf{Sat}(\Gamma)$,
- if $\forall R_t.C \in \mathsf{Sat}(\Gamma)$ then $[\mathbf{A}_t, I_t]C \in \mathsf{Sat}(\Gamma)$,
- if $[\mathbf{A}_t, Q]C \in \mathsf{Sat}(\Gamma)$ and $Q \cap F_t \neq \emptyset$ then $C \in \mathsf{Sat}(\Gamma)$. ◁

Thus, $\mathsf{Sat}(\Gamma)$ is a superset of Γ, which is equivalent to Γ.

Definition 7.3. The *transfer of Γ through R_t* is defined as follows:

$$\mathsf{Trans}(\Gamma, t) = \mathsf{Sat}(\{[\mathbf{A}_s, \delta_s(Q, t)]C \mid [\mathbf{A}_s, Q]C \in \Gamma\}).$$

The intuition behind $\mathsf{Trans}(\Gamma, t)$ is that, in a given interpretation, if Γ holds for x and $R_t(x, y)$ holds then $\mathsf{Trans}(\Gamma, t)$ holds for y.

Definition 7.4. The *compact form* $\mathsf{CF}(\Gamma)$ of Γ is the smallest set of formulas obtained as follows:

- if $C \in \Gamma$ and C is not of the form $[\mathbf{A}_t, Q]D$ then $C \in \mathsf{CF}(\Gamma)$,
- if $Q_1, \ldots, Q_k$ are all the sets such that $[\mathbf{A}_t, Q_i]C \in \Gamma$ for $1 \leq i \leq k$, then $[\mathbf{A}_t, Q_1 \cup \ldots \cup Q_k]C \in \mathsf{CF}(\Gamma)$. ◁

Function $\mathsf{Find}(\Gamma)$

1. if there exists $z \in \Delta \setminus \Delta_0$ with $\mathbf{C}(z) = \Gamma$ then return z,
2. else add a new element z to Δ with $\mathbf{C}(z) := \Gamma$ and return z.

Procedure $\mathsf{Simulate\text{-}Changing\text{-}Content}(x, \Gamma)$

1. $x_* := \mathsf{Find}(\Gamma)$;
2. for every y, t, C, if $\mathbf{E}(y, \exists R_t.C) = x$ then $\mathbf{E}(y, \exists R_t.C) := x_*$;
3. for every y and t, if $\mathbf{U}(y, R_t) = x$ then $\mathbf{U}(y, R_t) := x_*$;

(Note: if $x_* \neq x$ and $x \notin \Delta_0$ then x becomes unreachable from Δ_0 via paths using $(\mathcal{U}(R_t))_{t \in \mathrm{IND}}$.)

Algorithm 1

Input: $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$, a deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ in $\mathcal{R}eg$, and finite automata $(\mathbf{A}_t)_{t \in \mathrm{IND}}$ specifying $\mathcal{R}$.
Output: A least $\mathfrak{s}$-pseudo-model $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.

1. let Δ_0 be the set of all individuals occurring in $\mathcal{A}$;
   if $\Delta_0 = \emptyset$ then $\Delta_0 := \{\tau\}$;
   set $\Delta := \Delta_0$;
   for all individuals a occurring in $\mathcal{A}$, set $\mathcal{O}(a) := a$;
   for $x \in \Delta$, set $\mathbf{C}(x) := \mathsf{CF}(\mathsf{Sat}(\mathcal{T} \cup \{A \mid A(x) \in \mathcal{A}\}))$;
   set $\mathbf{E}$ and $\mathbf{U}$ to empty maps;
   set $\mathcal{T}' := \mathsf{CF}(\mathsf{Sat}(\mathcal{T}))$;
2. for every $x \in \Delta$ and every $C \in \mathbf{C}(x)$
   (a) case $C = (D \sqcap D')$:
      i. if $x \in \Delta_0$ then $\mathbf{C}(x) := \mathsf{CF}(\mathbf{C}(x) \cup \mathsf{Sat}(\{D, D'\}))$
      ii. else $\mathsf{Simulate\text{-}Changing\text{-}Content}(x, \mathsf{CF}(\mathbf{C}(x) \cup \mathsf{Sat}(\{D, D'\})))$;
   (b) case $C = (D \sqsubseteq D')$: if $\mathcal{I}, x \models_{\mathfrak{sc}} D$ then
      i. if $x \in \Delta_0$ then $\mathbf{C}(x) := \mathsf{CF}(\mathbf{C}(x) \cup \mathsf{Sat}(\{D'\}))$
      ii. else $\mathsf{Simulate\text{-}Changing\text{-}Content}(x, \mathsf{CF}(\mathbf{C}(x) \cup \mathsf{Sat}(\{D'\})))$;
   (c) case $C = \exists R_t.D$: if $\mathbf{E}(x, \exists R_t.D)$ is not defined then
      $\mathbf{E}(x, \exists R_t.D) := \mathsf{Find}(\mathsf{CF}(\mathsf{Sat}(\{D\}) \cup \mathsf{Trans}(\mathbf{C}(x), t) \cup \mathcal{T}'))$;
3. for every $x \in \Delta$ and every $t \in$ IND:
   (a) for every $y \in \Delta_0$ such that $\mathcal{U}_0(R_t)(x, y)$ holds:
      $\mathbf{C}(y) := \mathsf{CF}(\mathbf{C}(y) \cup \mathsf{Trans}(\mathbf{C}(x), t))$;
   (b) for every $y \in \Delta \setminus \Delta_0$ such that $\mathcal{U}_0(R_t)(x, y)$ holds:
      i. $y_* := \mathsf{Find}(\mathsf{CF}(\mathbf{C}(y) \cup \mathsf{Trans}(\mathbf{C}(x), t)))$;
      ii. for every D, if $\mathbf{E}(x, \exists R_t.D) = y$ then $\mathbf{E}(x, \exists R_t.D) := y_*$;
      iii. if $\mathbf{U}(x, R_t) = y$ then $\mathbf{U}(x, R_t) := y_*$;
   (c) if $\mathbf{U}(x, R_t)$ is not defined then
      $\mathbf{U}(x, R_t) := \mathsf{Find}(\mathsf{CF}(\mathsf{Trans}(\mathbf{C}(x), t) \cup \mathcal{T}'))$;
4. while some change occurred, go to Step 2;
5. for every $x \in \Delta \setminus \Delta_0$, if x is unreachable from Δ_0 in the sense that there do not exist a path $x_0 \in \Delta_0$, $x_1, \ldots, x_{k-1}$, $x_k = x$ and role names $R_{s_1}, \ldots, R_{s_k}$ such that $\mathcal{U}_0(R_{s_i})(x_{i-1}, x_i)$ holds for every $1 \leq i \leq k$
   then delete x from Δ and delete the elements of $\mathbf{E}$ and $\mathbf{U}$ that are related with x;

Fig. 2. Algorithm for constructing least pseudo-models.

Thus, $\mathsf{CF}(\Gamma)$ is a logical equivalent of Γ, which is more compact than Γ.

Algorithm 1 given in Figure 2 constructs a least $\mathfrak{s}$-pseudo-model for a deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ in $\mathcal{R}eg$. Our construction of a least pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ is done using the technique of building model graphs [27, 29, 31]. At the beginning, Δ starts from Δ_0, which consists of all the individuals occurring in $\mathcal{A}$ or some τ if $\mathcal{A}$ is empty, with $\mathbf{C}(x)$, for $x \in \Delta_0$, being the compact form of the saturation of $\mathcal{T} \cup \{A \mid A(x) \in \mathcal{A}\}$. Then for each $x \in \Delta$ and each formula $C \in \mathbf{C}(x)$, we "realize the requirement C at x" as follows:

1. Case $C = (D \sqcap D')$ (Step 2a) : If $x \in \Delta_0$ then we add both D and D' to $\mathbf{C}(x)$. Otherwise, to restrict the size of the constructed graph and prevent the situation when $\mathbf{C}(x)$ becomes equal to $\mathbf{C}(x')$ for some $x' \in \Delta \setminus \Delta_0$ different from x, we do not add D and D' to $\mathbf{C}(x)$ but "simulate the role of x" by $x_* \in \Delta \setminus \Delta_0$ with $\mathbf{C}(x_*) = \mathsf{CF}(\mathbf{C}(x) \cup \mathsf{Sat}(\{D, D'\}))$. The simulation is done by the procedure Simulate-Changing-Content, which replaces the connections to x by connections to x_* (by modifying the mappings $\mathbf{E}$ and $\mathbf{U}$).
2. Case $C = (D \sqsubseteq D')$ (Step 2b) : If D is *certainly satisfied* at x w.r.t. semantics $\mathfrak{s}$, denoted by $\mathcal{I}, x \models_{\mathfrak{s}c} D$, then analogously as for the above case, if $x \in \Delta_0$ then we add D' to $\mathbf{C}(x)$, else we simulate the role of x by x_* with $\mathbf{C}(x_*) = \mathsf{CF}(\mathbf{C}(x) \cup \mathsf{Sat}(\{D'\}))$ by calling the procedure Simulate-Changing-Content. Let us explain the satisfaction relation $\models_{\mathfrak{s}c}$. Consider the example case when $\mathcal{R} = \emptyset$, $\mathbf{C}(x) = \{D \sqsubseteq D', \exists R_t.A\}$ with $D = \forall\exists R_t.A$, $\mathbf{E}(x, \exists R_t.A)$ is defined, but $\mathbf{U}(x, R_t)$ is not. Then we may have that $\mathcal{I}, x \models_{\mathfrak{s}} \forall\exists R_t.A$ because there may be only one edge of R_t going out from x, which is undesirable since $\forall\exists R_t.A$ is not a "logical consequence" of $\exists R_t.A$. The problem is that $\mathbf{U}(x, R_t)$ is not yet defined. So, we define the satisfaction relation $\mathcal{I}, x \models_{\mathfrak{s}c} D$ as follows.

   Definition 7.5. Let $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ be a pseudo-interpretation. Define the satisfaction relation $\mathcal{I}, x \models_{\mathfrak{s}c} D$ for a positive allsome-formula D recursively as follows:

   $\mathcal{I}, x \models_{\mathfrak{s}c} \top$,
   $\mathcal{I}, x \models_{\mathfrak{s}c} A$ iff $x \in \mathcal{C}(A)$,
   $\mathcal{I}, x \models_{\mathfrak{s}c} D_1 \sqcap D_2$ iff $\mathcal{I}, x \models_{\mathfrak{s}c} D_1$ and $\mathcal{I}, x \models_{\mathfrak{s}c} D_2$,
   $\mathcal{I}, x \models_{\mathfrak{s}c} D_1 \sqcup D_2$ iff $\mathcal{I}, x \models_{\mathfrak{s}c} D_1$ or $\mathcal{I}, x \models_{\mathfrak{s}c} D_2$,
   $\mathcal{I}, x \models_{\mathfrak{s}c} \exists R_t.D$ iff $\exists y.(\mathcal{E}(R_t)(x, y) \wedge \mathcal{I}, y \models_{\mathfrak{s}c} D)$, and
   $\mathcal{I}, x \models_{\mathfrak{s}c} \forall\exists R_t.D$ iff $(*)$

   where $(*)$ is the combination of the following conditions:
   - $\forall y.(\mathcal{U}(R_t)(x, y) \to \mathcal{I}, y \models_{\mathfrak{s}c} D)$ and
   - case $\mathfrak{s} = \mathsf{Sem}_1$: $\mathcal{I}, x \models_{\mathfrak{s}c} \exists R_t.\top$ and $\mathbf{U}(x, R_t)$ is defined,
   - case $\mathfrak{s} = \mathsf{Sem}_{2,\mathcal{R}}$: for every consequence $R_{s_1} \circ \cdots \circ R_{s_k} \sqsubseteq R_t$ of $\mathcal{R}$ and for every path $x_0\,\mathcal{U}(R_{s_1})x_1 \ldots \mathcal{U}(R_{s_{i-1}})x_{i-1}$ with $x_0 = x$ and $1 \leq i \leq k$, $\mathcal{I}, x_{i-1} \models_{\mathfrak{s}c} \exists R_{s_i}.\top$ and $\mathbf{U}(x_{i-1}, R_{s_i})$ is defined. ◁

3. Case $C = \exists R_t.D$ (Step 2c) : If $\mathbf{E}(x, \exists R_t.D)$ is not defined, we just connect x via R_t to the possible world of $\Delta \setminus \Delta_0$ with content $\mathsf{CF}(\mathsf{Sat}(\{D\} \cup \mathsf{Trans}(\mathbf{C}(x), t) \cup \mathcal{T}))$ by setting $\mathbf{E}(x, \exists R_t.D)$ to that world. The world is created if necessary.

4. Case $C = [\mathbf{A}_s, Q]D$ (Steps 3a and 3b) : For every $y \in \Delta$ and $t \in \text{IND}$ such that $\mathcal{U}_0(R_t)(x, y)$ holds, we would like to add $[\mathbf{A}_s, \delta_s(Q, t)]D$ to $\mathbf{C}(y)$. More generally, for every $y \in \Delta$ and $t \in \text{IND}$, we would like to add $\mathsf{Trans}(\mathbf{C}(x), t)$ to $\mathbf{C}(y)$, independently from C. We do so for $y \in \Delta_0$ (because the edge $\mathcal{U}_0(R_t)(x, y)$ is caused by $R_t(x, y) \in \mathcal{A}$). For $y \in \Delta \setminus \Delta_0$, however, modifying the content of y has two drawbacks: First, other possible worlds connected to y will be affected. For example, if D is added to $\mathbf{C}(y)$ and $\mathcal{E}(R_{t'})(z, y)$ holds, then $\exists R_{t'}.D$ becomes satisfied at z, while x and z may be independent. Second, modifying $\mathbf{C}(y)$ may cause $\mathbf{C}(y) = \mathbf{C}(y')$ for some $y' \in \Delta \setminus \Delta_0$ different from y, which we try to avoid. Step 3b contains our solution for these two problems.
5. The case $C = \forall R_t.D$ is reduced to the case $C = [\mathbf{A}_t, I_t]D$.

In Step 3c of Algorithm 1, we also guarantee that for every $x \in \Delta$ and every $t \in \text{IND}$, x is connected via $\mathcal{U}(R_t)$ to the possible world with content $\mathsf{CF}(\mathsf{Sat}(\mathsf{Trans}(\mathbf{C}(x), t) \cup \mathcal{T}))$ by setting $\mathbf{U}(x, R_t)$ to that world. This is important for making $\mathcal{I}$ a least $\mathfrak{s}$-pseudo-model of the input knowledge base (in particular, for satisfying Condition 3 of the definition of $\leq_r$).

When iteration of Steps 2 and 3 does not modify the model graph anymore, we delete all possible worlds that are unreachable from Δ_0 (via a path using edges of $\mathcal{U}_0$). This is necessary because such a possible world x may contain a formula C which is not satisfied at x (for example, to realize $D \sqcap D' \in \mathbf{C}(x)$ we did not add D and D' to $\mathbf{C}(x)$ but just simulated the task).

Observe that, for any x, y, t, C, if $\mathbf{E}(x, \exists R_t.C) = y$ or $\mathbf{U}(x, R_t) = y$ then $y \notin \Delta_0$. Hence, if $\mathcal{U}_0(R_t)(x, y)$ holds and $y \in \Delta_0$, then we must have that $x \in \Delta_0$ and $R_t(x, y) \in \mathcal{A}$. Both Δ_0 and the part restricted to Δ_0 of $\mathcal{U}_0$ are therefore "fixed" by the ABox $\mathcal{A}$ (unless $\mathcal{A}$ is empty and $\Delta_0 = \{\tau\}$). This explains the way we deal with realizing $C \in \mathbf{C}(x)$ for $x \in \Delta_0$ in Steps 2(a)i, 2(b)i and 3a. On the other hand, the part on $\Delta \setminus \Delta_0$ of the constructed pseudo-model is flexible, and for that part we can "Simulate-Changing-Content" and redirect edges. Our techniques for that part prevent duplicates and avoid modifying the content of a node and re-creating a new node with that content later.

7.1 Illustrative Examples

Example 7.6. Consider the domain of web pages. Let

$$\begin{aligned}
\mathcal{R} &= \emptyset \\
\mathcal{T} &= \{\mathit{perfect} \sqsubseteq \mathit{interesting} \sqcap \forall \mathit{link}.\mathit{perfect}, \\
&\qquad \mathit{exists_link_to_perfect_page} \sqsubseteq \exists \mathit{link}.\mathit{perfect}\} \\
\mathcal{A} &= \{\mathit{perfect}(a), \mathit{exists_link_to_perfect_page}(b), \mathit{link}(a, b)\}
\end{aligned}$$

The empty RBox $\mathcal{R}$ is regular and specified by the following finite automaton, where $\text{IND} = \{l\}$ and R_l stands for *link*:

$$\mathbf{A}_l = \langle \text{IND}, \{1, 2\}, \{1\}, \{(1, l, 2)\}, \{2\} \rangle$$

As abbreviations, let

$$L = \mathit{link},\ I = \mathit{interesting},\ P = \mathit{perfect},\ E = \mathit{exists\_link\_to\_perfect\_page},$$

and let C and D denote the first and the second program clause of $\mathcal{T}$, respectively. In Figures 3 and 4 we illustrate the run of Algorithm 1 on the deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ using $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$. ◁

Example 7.7. In Figure 5, we present another example of application of Algorithm 1. ◁

7.2 Complexity and Correctness of the Algorithm

Proposition 7.8. *Let $\mathfrak{s}$ and $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ be inputs for Algorithm 1 and let $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ be the output. Assume that $\mathcal{R}$ and $\mathcal{T}$ are fixed, while $\mathcal{A}$ varies and has size n. Then Δ has size $O(n)$ and the algorithm runs in $O(n^4)$ steps.*

Proof. We will refer to the data structures used in Algorithm 1. Define the size of a set of formulas to be the sum of the lengths of its formulas.

Since each $x \in \Delta \setminus \Delta_0$ has a unique $\mathbf{C}(x)$ and the possible values of $\mathbf{C}(x)$ are dependent only on $\mathcal{R}$ and $\mathcal{T}$, the set $\Delta \setminus \Delta_0$ contains only $O(1)$ elements. Hence Δ has size $O(n)$.

Since $\mathcal{R}$ and $\mathcal{T}$ are fixed, the size of $\mathbf{C}(x)$ for $x \in \Delta \setminus \Delta_0$ and the size of $\mathbf{C}(a) \setminus \{A \mid A(a) \in \mathcal{A}\}$ for $a \in \Delta_0$ are bounded by a constant. Denote this assertion by $(*)$.

The total number of changes made at Steps 2(a)i, 2(b)i, 2c, 3a, 3c is of rank $O(n)$. Note that if x is "simulated" by x_* at Step 2(a)ii or 2(b)ii then $\mathbf{C}(x_*)$ extends $\mathbf{C}(x)$. A similar statement can be said for y and y_* at Step 3b. Since $\Delta \setminus \Delta_0$ has size $O(1)$, Δ has size $O(n)$, and $(*)$, the total number of changes made at Steps 2(a)ii, 2(b)ii, 3b is of rank $O(n)$. Hence, the loop at Step 4 executes only $O(n)$ times.

By $(*)$, all the calls of Sat, Trans, CF in the algorithm can be done in constant time. A call of Simulate-Changing-Content runs in time $O(n)$ (because y ranges over Δ). Similarly to the proof of Proposition 6.9, checking $\mathcal{I}, x \models_{\mathfrak{s}c} D$ at Step 2b can be done in time $O(n^3 \times \mathit{length}(D)) = O(n^3)$.

Summing up, Algorithm 1 runs in $O(n^4)$ steps. ◁

Here is the main theorem of this section:

Theorem 7.9. *Let $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ be a deterministic Horn knowledge base in $\mathcal{R}eg$, and $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ the pseudo-interpretation constructed by Algorithm 1 for $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ w.r.t. $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$. Then $\mathcal{I}$ is a least $\mathfrak{s}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.*

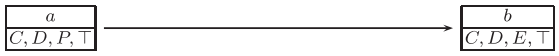
To prove this theorem we need a number of lemmas. The main ones are Lemma 7.13, which shows that $\mathcal{I}$ is an $\mathfrak{s}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, and Lemma 7.15, which shows that $\mathcal{I}$ is less than or equal (w.r.t. semantics $\mathfrak{s}$) to any $\mathfrak{s}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.

Lemma 7.10. *Algorithm 1 has the following properties:*

1. *During the execution, for every $x \in \Delta$ and every formula $\exists R_t.D$, if $\mathbf{E}(x, \exists R_t.D) = y$ then $\exists R_t.D \in \mathbf{C}(x)$ and $D \in \mathbf{C}(y)$.*
2. *At the end, $\mathbf{E}(x, \exists R_t.D)$ is defined for every $x \in \Delta$ and every formula $\exists R_t.D \in \mathbf{C}(x)$, and $\mathbf{U}(x, R_t)$ is defined for every $x \in \Delta$ and every role name R_t.*

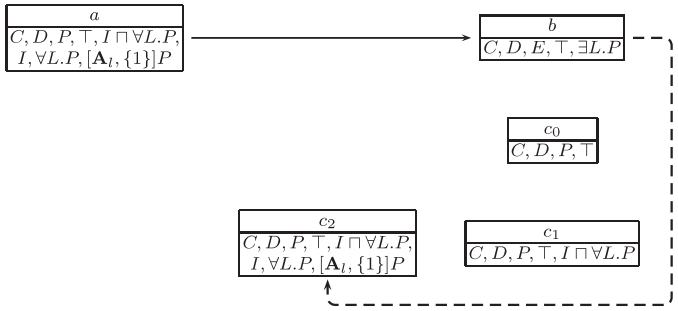
This lemma can easily be verified.

The model graph after executing Step 1:



The first execution of Step 2 results in the following model graph:

- Realizing C at $a : I \sqcap \forall L.P$ is added to $\mathbf{C}(a)$.
- Realizing $I \sqcap \forall L.P$ at $a : I$, $\forall L.P$, $[\mathbf{A}_l, \{1\}]P$ are added to $\mathbf{C}(a)$.
- Realizing D at $b : \exists L.P$ is added to $\mathbf{C}(b)$.
- Realizing $\exists L.P$ at $b : c_0$ is created and $\mathbf{E}(b, \exists L.P)$ is set to c_0.
- Realizing C at $c_0 : c_1$ is created and $\mathbf{E}(b, \exists L.P)$ is changed to c_1.
- Realizing $I \sqcap \forall L.P$ at $c_1 : c_2$ is created and $\mathbf{E}(b, \exists L.P)$ is changed to c_2.



The first execution of Step 3 results in the following model graph:

- Processing $a : [\mathbf{A}_l, \{2\}]P$ and P are added to $\mathbf{C}(b)$; d_0 is created and $\mathbf{U}(a, L)$ is set to d_0.
- Processing $b : e$ is created and $\mathbf{U}(b, L)$ is set to e.
- Processing $c_0 : \mathbf{U}(c_0, L)$ is set to e.
- Processing $c_1 : \mathbf{U}(c_1, L)$ is set to e.
- Processing $c_2 : \mathbf{U}(c_2, L)$ is set to d_0.
- Processing $d_0 : \mathbf{U}(d_0, L)$ is set to e.
- Processing $e : \mathbf{U}(e, L)$ is set to e.

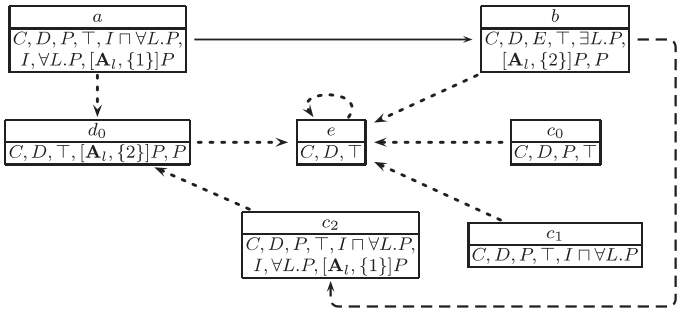
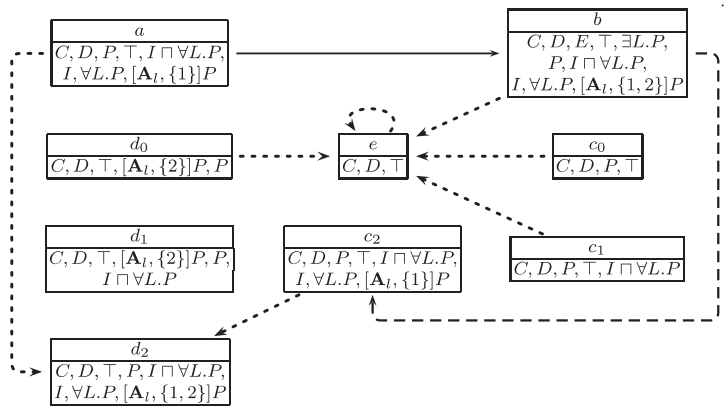


Fig. 3. The application of Algorithm 1 to $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, where: $L \stackrel{\text{def}}{=} R_l$ is the only role ($\text{IND} = \{l\}$), $\mathcal{R} = \emptyset$, and $\mathbf{A}_l = \langle \text{IND}, \{1, 2\}, \{1\}, \{(1, l, 2)\}, \{2\} \rangle$; $\mathcal{T} = \{C, D\}$ with $C \stackrel{\text{def}}{=} (P \sqsubseteq I \sqcap \forall L.P)$ and $D \stackrel{\text{def}}{=} (E \sqsubseteq \exists L.P)$; and $\mathcal{A} = \{P(a), E(b), L(a, b)\}$. Each node of the model graph represents an element of Δ (displayed in the upper part of the node). The lower part of a node x contains the formulas of $\mathbf{C}(x)$. We have $\Delta_0 = \{a, b\}$. The dashed (resp. dotted) arrows represent elements of $\mathbf{E}$ (resp. $\mathbf{U}$). To be continued in Fig. 4.

The second execution of Step 2 (looping at Step 4) results in the following model graph:

- Realizing C at $b : I \sqcap \forall L.P$ is added to $\mathbf{C}(b)$.
- Realizing $I \sqcap \forall L.P$ at $b : I$ and $\forall L.P$ are added to $\mathbf{C}(b)$, and
  $[\mathbf{A}_l, \{2\}]P$ is replaced by $[\mathbf{A}_l, \{1, 2\}]P$.
- Realizing C at $d_0 : d_1$ is created, $\mathbf{U}(a, d_0)$ and $\mathbf{U}(c_2, d_0)$ are changed to d_1.
- Realizing $I \sqcap \forall L.P$ at $d_1 : d_2$ is created, $\mathbf{U}(a, L)$ and $\mathbf{U}(c_2, L)$ are changed to d_2.



The second execution of Step 3 and the third execution of Step 2 (looping at Step 4) results in the following model graph:

- Processing b (at Step 3) : $\mathbf{E}(b, \exists L.P)$ is changed to d_2; $\mathbf{U}(b, L)$ is changed to d_0.
- Processing d_1 (at Step 3) : $\mathbf{U}(d_1, L)$ is set to e.
- Processing d_2 (at Step 3) : $\mathbf{U}(d_2, L)$ is set to d_0.
- Realizing C at d_0 (at Step 2) : $\mathbf{U}(b, L)$ and $\mathbf{U}(d_2, L)$ are changed to d_1.
- Realizing $I \sqcap \forall L.P$ at d_1 (at Step 2) : $\mathbf{U}(b, L)$ and $\mathbf{U}(d_2, L)$ are changed to d_2.

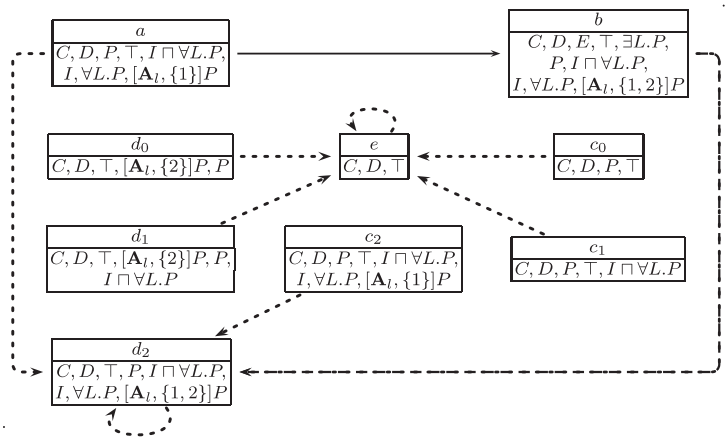


Fig. 4. Continuation of Figure 3. The dashed and dotted arrow (b, d_2) in the second model graph represents the connections $\mathbf{E}(b, \exists L.P) = d_2$ and $\mathbf{U}(b, L) = d_2$. The second model graph is not further modified by the loop at Step 4. Executing Step 5, the nodes c_0, c_1, c_2, d_0, d_1, e and the related arrows are deleted.

In this figure, we present the model graph constructed by Algorithm 1 for the deterministic Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ of the example given in Section 5.2, using semantics $\mathfrak{s} = \mathsf{Sem}_1$. Recall that:

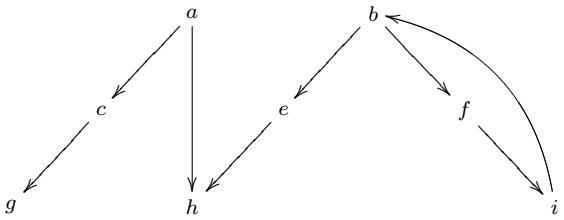
- The regular RBox $\mathcal{R} = \{link \sqsubseteq path, link \circ path \sqsubseteq path\}$ is specified by the following finite automata, where $\mathrm{IND} = \{l, p\}$, with R_l standing for $link$ and R_p standing for $path$:

$$\mathbf{A}_l = \langle \mathrm{IND}, \{1,2\}, \{1\}, \{(1,l,2)\}, \{2\}\rangle$$
$$\mathbf{A}_p = \langle \mathrm{IND}, \{1,2\}, \{1\}, \{(1,l,1),(1,l,2),(1,p,2)\}, \{2\}\rangle.$$

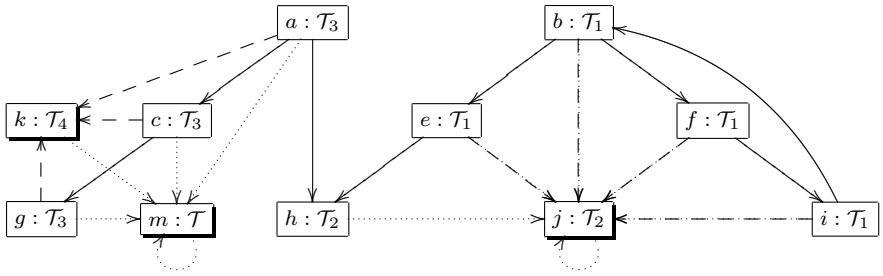
- The TBox $\mathcal{T}$ consists of the following deterministic program clauses:

$$perfect \sqsubseteq interesting \sqcap \forall path.interesting$$
$$interesting \sqcap \forall \exists path.interesting \sqsubseteq perfect$$
$$interesting \sqcup \forall \exists link.interesting \sqsubseteq worth\_surfing$$
$$\exists path.interesting \sqsubseteq has\_a\_path\_to\_an\_interesting\_page$$
$$has\_a\_path\_to\_an\_interesting\_page \sqsubseteq \exists path.interesting.$$

- The ABox $\mathcal{A}$ consists of $perfect(b)$, $has\_a\_path\_to\_an\_interesting\_page(g)$ and the following role assertions of $link$:



The constructed model graph is illustrated below. Each $x \in \Delta$ is represented by a box labeled by $x : \mathbf{C}(x)$. The box is shaded if $x \in \Delta \setminus \Delta_0$. A normal arrow from x to y represents the role assertion $link(x, y) \in \mathcal{A}$. A dotted arrow from x to y represents the two connections $\mathbf{U}(x, link) = y$ and $\mathbf{U}(x, path) = y$. A dashed arrow from x to y represents the connection $\mathbf{E}(x, \exists path.interesting) = y$. A dashed and dotted arrow (like the one from b to j) is a combination of a dashed arrow and a dotted arrow. We do not display all arrows of $path$.



$$\mathcal{T}_1 = \mathcal{T} \cup \{perfect, interesting \sqcap \forall path.interesting, interesting,$$
$$\forall path.interesting, [\mathbf{A}_p, \{1,2\}]interesting, worth\_surfing,$$
$$has\_a\_path\_to\_an\_interesting\_page, \exists path.interesting\}$$
$$\mathcal{T}_2 = \mathcal{T} \cup \{[\mathbf{A}_p, \{1,2\}]interesting, interesting, worth\_surfing\}$$
$$\mathcal{T}_3 = \mathcal{T} \cup \{has\_a\_path\_to\_an\_interesting\_page, \exists path.interesting\}$$
$$\mathcal{T}_4 = \mathcal{T} \cup \{interesting, worth\_surfing\}$$

Fig. 5. An example of application of Algorithm 1.

Lemma 7.11. *Consider a moment in an execution of Algorithm 1. Suppose that $\mathcal{E}(R_t)(x,y)$ (resp. $\mathcal{U}(R_t)(x,y)$) holds. Then there exist $x_0, \ldots, x_k$ in Δ with $x_0 = x$, $x_k = y$ and indices $s_1, \ldots, s_k \in \mathrm{IND}$ such that $\mathcal{E}_0(R_{s_i})(x_{i-1}, x_i)$ (resp. $\mathcal{U}_0(R_{s_i})(x_{i-1}, x_i)$) holds for $1 \leq i \leq k$ and $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ is a consequence of the RBox $\mathcal{R}$.*

Proof. By induction on the number of inferences in the derivation of $\mathcal{E}(R_t)(x,y)$ (resp. $\mathcal{U}(R_t)(x,y)$) when extending $(\mathcal{E}_0(R_t))_{t\in\mathrm{IND}}$ to $(\mathcal{E}(R_t))_{t\in\mathrm{IND}}$ (resp. $(\mathcal{U}_0(R_t))_{t\in\mathrm{IND}}$ to $(\mathcal{U}(R_t))_{t\in\mathrm{IND}}$) using the RBox $\mathcal{R}$. ◁

Lemma 7.12. *The following conditions are equivalent for every regular RBox $\mathcal{R}$:*

1. *$s_1 \ldots s_k$ is derivable from t using the grammar $G(\mathcal{R})$,*
2. *the role axiom $R_{s_1} \circ \cdots \circ R_{s_k} \sqsubseteq R_t$ is a consequence of $\mathcal{R}$.*

Proof sketch. The first assertion clearly implies the second one. The converse can be proved by induction using a model of the form of an infinite full tree. See also [3, 8, 15]. ◁

Lemma 7.13. *Let $\mathfrak{s}$ and $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ be inputs and $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U}\rangle$ the output of Algorithm 1. Then $\mathcal{I}$ is an $\mathfrak{s}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. Moreover, for every $x \in \Delta$ and every $C \in \mathbf{C}(x)$, we have that $\mathcal{I}, x \models_{\mathfrak{s}} C$, where $\mathbf{C}$ is the data structure used by Algorithm 1 for $\mathfrak{s}$ and $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.*

Proof. For every individual a, we have that $\mathcal{O}(a) = a \in \Delta_0$. By Step 1, $\mathcal{I} \models A(a)$ for every $A(a) \in \mathcal{A}$. By the definition of $\mathcal{E}_0$, $\mathcal{I}$ also satisfies every assertion $R(a,b) \in \mathcal{A}$. That is, $\mathcal{I}$ is a pseudo-model of $\mathcal{A}$. By the definition of $\mathcal{E}$ and $\mathcal{U}$, $\mathcal{I}$ is a pseudo-model of $\mathcal{R}$. Since $\mathcal{T}$ is included in the content of every possible world, to show that $\mathcal{I}$ is an $\mathfrak{s}$-pseudo-model of $\mathcal{T}$ it suffices to show that, for every $x \in \Delta$ and every $C \in \mathbf{C}(x)$, it holds that $\mathcal{I}, x \models_{\mathfrak{s}} C$. We prove this by induction on the construction of C.

Consider the case when C is of the form $D \sqsubseteq D'$. Suppose that $\mathcal{I}, x \models_{\mathfrak{s}} D$. We show that $\mathcal{I}, x \models_{\mathfrak{s}} D'$. Since $\mathbf{U}(x, R_t)$ is defined for every $x \in \Delta$ and every role name R_t, we have that $\mathcal{I}, x \models_{\mathfrak{s}c} D$ iff $\mathcal{I}, x \models_{\mathfrak{s}} D$. Hence $\mathcal{I}, x \models_{\mathfrak{s}c} D$. If $x \in \Delta_0$, then D' must have been added to $\mathbf{C}(x)$, and by the inductive assumption, $\mathcal{I}, x \models_{\mathfrak{s}} D'$. Suppose that $x \in \Delta \setminus \Delta_0$. When Step 2b is executed the last time for x and C, as no changes are made, we have $x_* = x$, where x_* is the element simulating the role of x, because x is reachable from Δ_0 via a path using $\mathcal{U}_0$ as it remains after executing Step 5. Since $D \sqsubseteq D' \in \mathbf{C}(x)$ and $\mathcal{I}, x \models_{\mathfrak{s}c} D$, we have that $D' \in \mathbf{C}(x_*)$, i.e. $D' \in \mathbf{C}(x)$. By the inductive assumption, $\mathcal{I}, x \models_{\mathfrak{s}} D'$. Therefore, $\mathcal{I}, x \models_{\mathfrak{s}} D \sqsubseteq D'$.

The case when C is of the form $D \sqcap D'$ is similar to the above case. The cases when C is of the form A or $\exists R_t.D$ are straightforward.

Consider the case when $C = \forall R_t.D$. Suppose that $\mathcal{U}(R_t)(x,y)$ holds. By the inductive assumption, it is sufficient to show that $D \in \mathbf{C}(y)$. Since $\mathcal{U}(R_t)(x,y)$ holds, by Lemma 7.11, there exist $x_0, \ldots, x_k$ in Δ with $x_0 = x$, $x_k = y$ and indices $s_1, \ldots, s_k \in \mathrm{IND}$ such that $\mathcal{U}_0(R_{s_i})(x_{i-1}, x_i)$ holds for $1 \leq i \leq k$ and $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ is a consequence of $\mathcal{R}$. By Lemma 7.12, $s_1 \ldots s_k$ is accepted by $\mathbf{A}_t$. Hence $\widetilde{\delta}_t(I_t, s_1 \ldots s_k) \cap F_t \neq \emptyset$. Since $\forall R_t.D \in \mathbf{C}(x)$, we have $[\mathbf{A}_t, Q]D \in \mathbf{C}(x_0)$ for some $Q \supseteq I_t$, and hence $[\mathbf{A}_t, Q']D \in \mathbf{C}(x_k)$ for some $Q' \supseteq \widetilde{\delta}_t(I_t, s_1 \ldots s_k)$. It follows that $D \in \mathbf{C}(x_k)$, which means $D \in \mathbf{C}(y)$.

Consider the case when $C = [\mathbf{A}_t, Q]D$. Let $x_0 \mathcal{U}(R_{s_1}) x_1 \ldots x_{k-1} \mathcal{U}(R_{s_k}) x_k$ be an arbitrary path of $\mathcal{I}$ such that $x_0 = x$ and $\widetilde{\delta}_t(Q, s_1 \ldots s_k) \cap F_t \neq \emptyset$. We show that $D \in \mathbf{C}(x_k)$.

There exists $q \in Q$ such that $\widetilde{\delta}_t(\{q\}, s_1 \dots s_k) \cap F_t \neq \emptyset$. Since $[\mathbf{A}_t, Q]D \in \mathbf{C}(x)$ and $q \in Q$, there exists $x'_0, \dots, x'_h \in \Delta$ and $s'_1, \dots, s'_h \in \mathrm{IND}$ such that $\forall R_t.D \in \mathbf{C}(x'_0)$, $q \in \widetilde{\delta}_t(I_t, s'_1 \dots s'_h)$, $x'_0 \mathcal{U}_0(R_{s'_1}) x'_1 \dots x'_{h-1} \mathcal{U}_0(R_{s'_h}) x'_h$ holds, and $x'_h = x = x_0$. The word $s'_1 \dots s'_h s_1 \dots s_k$ is thus accepted by $\mathbf{A}_t$. By Lemma 7.12, it follows that $R_{s'_1} \circ \dots \circ R_{s'_h} \circ R_{s_1} \circ \dots \circ R_{s_k} \sqsubseteq R_t$ is a consequence of $\mathcal{R}$. Hence $\mathcal{U}(R_t)(x'_0, x_k)$ holds. Since $\forall R_t.D \in \mathbf{C}(x'_0)$ and $\mathcal{U}(R_t)(x'_0, x_k)$ holds, as for the previous case (when C is of the form $\forall R_t.D$) we derive that $D \in \mathbf{C}(x_k)$. By the inductive assumption, it follows that $\mathcal{I}, x_k \models_{\mathfrak{s}} D$. Hence $\mathcal{I}, x \models_{\mathfrak{s}} C$, which completes the proof. ◁

Lemma 7.14. *Let $\mathfrak{s}$ and $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ be inputs and $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ the output of Algorithm 1, and let $\mathcal{I}' = \langle \Delta', \mathcal{O}', \mathcal{C}', \mathcal{E}', \mathcal{U}' \rangle$ be an arbitrary $\mathfrak{s}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. Consider a moment after executing a numerated step in the execution of Algorithm 1. Let*

$$r = \{(a, a^{\mathcal{I}'}) \mid a \text{ is an individual occurring in } \mathcal{A}\} \cup \{(x, x') \in \Delta \times \Delta' \mid x \text{ is not an individual and } \mathcal{I}', x' \models_{\mathfrak{s}} \mathbf{C}(x)\}$$

Then the following assertions hold for every $u, v \in \Delta$, $u', v' \in \Delta'$, $s \in \mathrm{IND}$, every formula D'' and every individual a occurring in $\mathcal{A}$:

$$r(u, u') \wedge (\mathbf{E}(u, \exists R_s.D'') = v) \wedge \mathcal{E}'(R_s)(u', v') \wedge (\mathcal{I}', v' \models_{\mathfrak{s}} D'') \rightarrow r(v, v') \quad (36)$$

$$r(u, u') \wedge (\mathbf{U}(u, R_s) = v) \wedge \mathcal{U}'(R_s)(u', v') \rightarrow r(v, v') \quad (37)$$

$$\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} \mathbf{C}(a) \quad (38)$$

Proof. We prove this lemma by induction on the number of executed steps. The base case occurs after executing Step 1 and the assertions clearly hold. Consider some latter enumerated step K of the algorithm. Inductively assume that the assertions (36)-(38) of the lemma hold before executing that step. We first prove the following claim by an inner induction on the construction of C_0.

Claim 1. *Let C_0 be a positive allsome-formula and suppose that $r(x, x')$ holds and $\mathcal{I}, x \models_{\mathfrak{s}c} C_0$. Then $\mathcal{I}', x' \models_{\mathfrak{s}} C_0$.*

- The cases when C_0 is of the form A or $D_1 \sqcap D_2$ or $D_1 \sqcup D_2$ are straightforward.
- Consider the case when $C_0 = \exists R_s.D_0$. Since $\mathcal{I}, x \models_{\mathfrak{s}c} C_0$, there exists y such that $\mathcal{E}(R_s)(x, y)$ holds and $\mathcal{I}, y \models_{\mathfrak{s}c} D_0$. By Lemma 7.11, there exist $x_0, \dots, x_k$ in Δ with $x_0 = x$, $x_k = y$ and indices $s_1, \dots, s_k \in \mathrm{IND}$ such that $\mathcal{E}_0(R_{s_i})(x_{i-1}, x_i)$ holds for $1 \leq i \leq k$ and $R_{s_1} \circ \dots \circ R_{s_k} \sqsubseteq R_s$ is a consequence of $\mathcal{R}$. Denote this by $(\star)$. For $1 \leq i \leq k$, since $\mathcal{E}_0(R_{s_i})(x_{i-1}, x_i)$ holds, either $R_{s_i}(x_{i-1}, x_i) \in \mathcal{A}$ or $\mathbf{E}(x_{i-1}, \exists R_{s_i}.C_i) = x_i$ for some C_i. Let $x'_0 = x'$. For i from 1 to k, we define $x'_i \in \Delta'$ as follows:
  - Case $R_{s_i}(x_{i-1}, x_i) \in \mathcal{A}$: We have that x_{i-1} and x_i are individuals, and by the definition of r, $x'_{i-1} = \mathcal{O}'(x_{i-1})$. Let $x'_i = \mathcal{O}'(x_i)$. Thus $r(x_i, x'_i)$ holds. Since $\mathcal{I}'$ is a pseudo-model of $\mathcal{A}$, $\mathcal{E}'(R_{s_i})(x'_{i-1}, x'_i)$ holds.
  - Case $\mathbf{E}(x_{i-1}, \exists R_{s_i}.C_i) = x_i$: We have that $\exists R_{s_i}.C_i \in \mathbf{C}(x_{i-1})$. Since $r(x_{i-1}, x'_{i-1})$ holds (we are maintaining this invariant), by the definition of r and the outer inductive assumption (38), we have that $\mathcal{I}', x'_{i-1} \models_{\mathfrak{s}} \exists R_{s_i}.C_i$. Hence, there exists $x'_i \in \Delta'$ such that $\mathcal{E}'(R_{s_i})(x'_{i-1}, x'_i)$ holds and $\mathcal{I}', x'_i \models_{\mathfrak{s}} C_i$. By the outer inductive assumption (36), $r(x_i, x'_i)$ holds.

We have that $r(x_k, x'_k)$ holds, and for $1 \leq i \leq k$, $\mathcal{E}'(R_{s_i})(x'_{i-1}, x'_i)$ holds. By $(\star)$, it follows that $\mathcal{E}'(R_s)(x'_0, x'_k)$ holds, which means that $\mathcal{E}'(R_s)(x', x'_k)$ holds. Since $r(x_k, x'_k)$ holds and $x_k = y$ and $\mathcal{I}, y \models_{\mathfrak{s}c} D_0$, by the inner inductive assumption, $\mathcal{I}', x'_k \models_{\mathfrak{s}} D_0$. It follows that $\mathcal{I}', x' \models_{\mathfrak{s}} \exists R_s.D_0$, i.e. $\mathcal{I}', x' \models_{\mathfrak{s}} C_0$.

– Now consider the case when $C_0 = \forall\exists R_s.D_0$. Let $y = \mathbf{U}(x, R_s)$. Thus $\mathcal{I}, y \models_{\mathfrak{s}c} D_0$. Let y' be an arbitrary element of Δ' such that $\mathcal{U}'(R_s)(x', y')$ holds. By the outer inductive assumption (37), we derive that $r(y, y')$ holds. By the inner inductive assumption, it follows that $\mathcal{I}', y' \models_{\mathfrak{s}} D_0$. Hence $\mathcal{I}', x' \models_{\mathfrak{s}} \forall R_s.D_0$.
Consider the case $\mathfrak{s} = \mathsf{Sem}_1$. Since $\mathcal{I}, x \models_{\mathfrak{s}c} C_0$, we have that $\mathcal{I}, x \models_{\mathfrak{s}c} \exists R_s.\top$. As proved for the case $C_0 = \exists R_s.D_0$, it follows that $\mathcal{I}', x' \models_{\mathfrak{s}} \exists R_s.\top$.
Now consider the case $\mathfrak{s} = \mathsf{Sem}_{2,\mathcal{R}}$. Let $R_{s_1} \circ \cdots \circ R_{s_k} \sqsubseteq R_s$ be a consequence of $\mathcal{R}$ and let $1 \leq i \leq k$. We show that $\mathcal{I}', x' \models_{\mathfrak{s}} \forall R_{s_1} \ldots \forall R_{s_{i-1}} \exists R_{s_i} \top$. Let $x'_0 = x'$ and $x'_1, \ldots, x'_{i-1}$ be arbitrary elements of Δ' such that $\mathcal{U}'(R_{s_j})(x'_{j-1}, x'_j)$ holds for $1 \leq j \leq i-1$. We need to show that $\mathcal{I}', x'_{i-1} \models_{\mathfrak{s}} \exists R_{s_i}.\top$. For $1 \leq j \leq i-1$, let $x_j = \mathbf{U}(x_{j-1}, R_{s_j})$, then, by the outer inductive assumption (37), $r(x_j, x'_j)$ holds. Since $\mathcal{I}, x \models_{\mathfrak{s}c} C_0$, we have that $\mathcal{I}, x_{i-1} \models_{\mathfrak{s}c} \exists R_{s_i}.\top$. As proved for the case $C_0 = \exists R_s.D_0$, it follows that $\mathcal{I}', x'_{i-1} \models_{\mathfrak{s}} \exists R_{s_i}.\top$. This completes the proof of Claim 1.

We now return to the outer induction. Suppose that after executing the step K we have r_2, Δ_2, $\mathbf{C}_2$, $\mathbf{E}_2$, $\mathbf{U}_2$, $\mathcal{I}_2$ in the places of r, Δ, $\mathbf{C}$, $\mathbf{E}$, $\mathbf{U}$, $\mathcal{I}$, respectively. We show that the following conditions hold for every $u, v \in \Delta$, $u', v' \in \Delta'$, $s \in \mathrm{IND}$, every formula D'' and every individual a occurring in $\mathcal{A}$:

$$r_2(u, u') \wedge (\mathbf{E}_2(u, \exists R_s.D'') = v) \wedge \mathcal{E}'(R_s)(u', v') \wedge (\mathcal{I}', v' \models_{\mathfrak{s}} D'') \rightarrow r_2(v, v') \quad (39)$$

$$r_2(u, u') \wedge (\mathbf{U}_2(u, R_s) = v) \wedge \mathcal{U}'(R_s)(u', v') \rightarrow r_2(v, v') \quad (40)$$

$$\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} \mathbf{C}_2(a) \quad (41)$$

– Consider the case $K = 2(a)i$ or $K = 2(b)i$. Since $r_2 = r$ and $\mathbf{E}_2 = \mathbf{E}$ and $\mathbf{U}_2 = \mathbf{U}$, the assertions (39) and (40) follow from the inductive assumptions (36) and (37). The assertion (41) clearly holds for the case $K = 2(a)i$. Consider the assertion (41) for the case $K = 2(b)i$ with $x = a$. It suffices to show that $\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} D'$. Since $\mathcal{I}, a \models_{\mathfrak{s}c} D$, by Claim 1, $\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} D$. By the inductive assumption, $\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} \mathbf{C}(a)$, hence $\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} (D \sqsubseteq D')$. It follows that $\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} D'$.
– Consider the case $K = 2(a)ii$. Let x' be an element of Δ' such that $r(x, x')$ holds. It suffices to show that $r_2(x_*, x')$ holds. Since $r(x, x')$ holds, $\mathcal{I}', x' \models_{\mathfrak{s}} \mathbf{C}(x)$, and we have that $\mathcal{I}', x' \models_{\mathfrak{s}} D \sqcap D'$. Hence $\mathcal{I}', x' \models_{\mathfrak{s}} D$ and $\mathcal{I}', x' \models_{\mathfrak{s}} D'$. It follows that $\mathcal{I}', x' \models_{\mathfrak{s}} \mathbf{C}(x_*)$, and hence $r_2(x, x_*)$ holds.
– Consider the case $K = 2(b)ii$ and assume that $\mathcal{I}, x \models_{\mathfrak{s}c} D$. Let x' be an element of Δ' such that $r(x, x')$ holds. It suffices to show that $r_2(x_*, x')$ holds. We need only to show that $\mathcal{I}', x' \models_{\mathfrak{s}} D'$. Since $r(x, x')$ holds, $\mathcal{I}', x' \models_{\mathfrak{s}} (D \sqsubseteq D')$. Since $\mathcal{I}, x \models_{\mathfrak{s}c} D$ and $r(x, x')$ holds, by Claim 1, $\mathcal{I}', x' \models_{\mathfrak{s}} D$. It follows that $\mathcal{I}', x' \models_{\mathfrak{s}} D'$.
– Consider the case $K = 2c$ and assume that $\mathbf{E}(x, \exists R_t.D)$ is not defined. It suffices to prove the assertion (39) for the case when $u = x$, $s = t$ and $D'' = D$. Consider this case. Suppose that $r_2(x, u') \wedge (\mathbf{E}_2(x, \exists R_t.D) = v) \wedge \mathcal{E}'(R_t)(u', v') \wedge (\mathcal{I}', v' \models_{\mathfrak{s}} D)$ holds. We show that $r_2(v, v')$ holds. Since $r_2(x, u')$ holds, $r(x, u')$ also holds, and hence $\mathcal{I}', u' \models_{\mathfrak{s}} \mathbf{C}(x)$. It follows that $\mathcal{I}', v' \models_{\mathfrak{s}} \mathsf{Trans}(\mathbf{C}(x), t)$, since $\mathcal{E}'(R_t)(u', v')$ holds. Hence

$\mathcal{I}', v' \models_{\mathfrak{s}} \{D\} \cup \mathsf{Trans}(\mathbf{C}(x), t) \cup \mathcal{T}$, which implies $\mathcal{I}', v' \models_{\mathfrak{s}} \mathbf{C}_2(v)$ (since $\mathcal{I}', v' \models_{\mathfrak{s}} \Gamma$ iff $\mathcal{I}', v' \models_{\mathfrak{s}} \mathsf{CF}(\mathsf{Sat}(\Gamma))$ for any set Γ of formulas), and hence $r_2(v, v')$ holds.

- Consider the case $K = 3a$. Since $y \in \Delta_0$, we have that $r_2 = r$. Additionally, $\mathbf{E}_2 = \mathbf{E}$ and $\mathbf{U}_2 = \mathbf{U}$, hence the assertions (39) and (40) follow from the inductive assumptions (36) and (37). For the assertion (41), it suffices to consider the case $y = a$ and show that $\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} \mathsf{Trans}(\mathbf{C}(x), t)$. Since $\mathcal{U}_0(R_t)(x, a)$ holds, x must be an individual occurring in $\mathcal{A}$ and $R_t(x, a) \in \mathcal{A}$. By the inductive assumption (38), $\mathcal{I}', x^{\mathcal{I}'} \models_{\mathfrak{s}} \mathbf{C}(x)$. Since $\mathcal{I}'$ is a pseudo-model of $\mathcal{A}$ and $R_t(x, a) \in \mathcal{A}$, it follows that $\mathcal{I}', a^{\mathcal{I}'} \models_{\mathfrak{s}} \mathsf{Trans}(\mathbf{C}(x), t)$.
- Consider the case $K = 3(b)ii$. It suffices to show that if $r(x, x') \wedge (\mathbf{E}(x, \exists R_t.D) = y) \wedge \mathcal{E}'(R_t)(x', y') \wedge (\mathcal{I}', y' \models_{\mathfrak{s}} D)$ then $\mathcal{I}', y' \models_{\mathfrak{s}} \mathbf{C}(y_*)$. Suppose that the premise holds. By the inductive assumption, $r(y, y')$ holds and $\mathcal{I}', y' \models_{\mathfrak{s}} \mathbf{C}(y)$. Since $r(x, x')$ holds, $\mathcal{I}', x' \models_{\mathfrak{s}} \mathbf{C}(x)$. Hence $\mathcal{I}', y' \models_{\mathfrak{s}} \mathsf{Trans}(\mathbf{C}(x), t)$ (since $\mathcal{E}'(R_t)(x', y')$ holds). Hence $\mathcal{I}', y' \models_{\mathfrak{s}} \mathbf{C}(y_*)$.
- Consider the case $K = 3(b)iii$. It suffices to show that if $r(x, x') \wedge (\mathbf{U}(x, R_t) = y) \wedge \mathcal{U}'(R_t)(x', y')$ holds then $\mathcal{I}', y' \models_{\mathfrak{s}} \mathbf{C}(y_*)$. This can be proved analogously as for Step 3(b)ii.
- Consider the case $K = 3c$ and assume that $\mathbf{U}(x, R_t)$ is not defined. It suffices to prove the assertion (40) for the case $u = x$ and $s = t$. Consider this case. Suppose that $r_2(x, u') \wedge (\mathbf{U}_2(x, R_t) = v) \wedge \mathcal{U}'(R_t)(u', v')$ holds. We show that $r_2(v, v')$ holds. Since $r_2(x, u')$ holds, $r(x, u')$ also holds, and hence $\mathcal{I}', u' \models_{\mathfrak{s}} \mathbf{C}(x)$. It follows that $\mathcal{I}', v' \models_{\mathfrak{s}} \mathsf{Trans}(\mathbf{C}(x), t)$, since $\mathcal{U}'(R_t)(u', v')$ holds. Hence $\mathcal{I}', v' \models_{\mathfrak{s}} \mathsf{Trans}(\mathbf{C}(x), t) \cup \mathcal{T}$, which means $\mathcal{I}', v' \models_{\mathfrak{s}} \mathbf{C}_2(v)$, and hence $r_2(v, v')$ holds.
- The case when $K = 5$ is trivial. ◁

Lemma 7.15. *Let $\mathcal{R}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{I}$, $\mathcal{I}'$ be as in Lemma 7.14, and r be the relation defined as in Lemma 7.14 for the end of the execution of Algorithm 1. Then $\mathcal{I} \leq_r \mathcal{I}'$.*

Proof. We check the four conditions of $\mathcal{I} \leq_r \mathcal{I}'$:

1. By the definition of r, for every a occurring in $\mathcal{A}$, $r(a^{\mathcal{I}}, a^{\mathcal{I}'})$ holds.
2. Suppose that $\mathcal{E}(R_s)(x, y) \wedge r(x, x')$ holds. We show that there exists $y' \in \Delta'$ such that $\mathcal{E}'(R_s)(x', y') \wedge r(y, y')$. Since $\mathcal{E}(R_s)(x, y)$ holds, by Lemma 7.11, there exist $x_0, \ldots, x_k$ in Δ with $x_0 = x$, $x_k = y$ and indices $s_1, \ldots, s_k \in \mathrm{IND}$ such that $\mathcal{E}_0(R_{s_i})(x_{i-1}, x_i)$ holds for $1 \leq i \leq k$ and $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_s$ is a consequence of $\mathcal{R}$. Denote this by (‡). For $1 \leq i \leq k$, since $\mathcal{E}_0(R_{s_i})(x_{i-1}, x_i)$ holds, either $R_{s_i}(x_{i-1}, x_i) \in \mathcal{A}$ or $\mathbf{E}(x_{i-1}, \exists R_{s_i}.C_i) = x_i$ for some C_i. Let $x'_0 = x'$. For i from 1 to k, we define $x'_i \in \Delta'$ as follows:
   - Case $R_{s_i}(x_{i-1}, x_i) \in \mathcal{A}$: We have that x_{i-1} and x_i are individuals, and by the definition of r, $x'_{i-1} = \mathcal{O}'(x_{i-1})$. Let $x'_i = \mathcal{O}'(x_i)$. Thus $r(x_i, x'_i)$ holds. Since $\mathcal{I}'$ is a pseudo-model of $\mathcal{A}$, $\mathcal{E}'(R_{s_i})(x'_{i-1}, x'_i)$ holds.
   - Case $\mathbf{E}(x_{i-1}, \exists R_{s_i}.C_i) = x_i$: We have that $\exists R_{s_i}.C_i \in \mathbf{C}(x_{i-1})$. Since $r(x_{i-1}, x'_{i-1})$ holds (we are maintaining this invariant), $\mathcal{I}', x'_{i-1} \models_{\mathfrak{s}} \exists R_{s_i}.C_i$. Hence, there exists $x'_i \in \Delta'$ such that $\mathcal{E}'(R_{s_i})(x'_{i-1}, x'_i)$ holds and $\mathcal{I}', x'_i \models_{\mathfrak{s}} C_i$. By Lemma 7.14, $r(x_i, x'_i)$ holds.

   We have that $r(x_k, x'_k)$ holds, and for $1 \leq i \leq k$, $\mathcal{E}'(R_{s_i})(x'_{i-1}, x'_i)$ holds. By (‡), it follows that $\mathcal{E}'(R_s)(x'_0, x'_k)$ holds. Take $y' = x'_k$. Thus $\mathcal{E}'(R_s)(x', y')$ and $r(y, y')$ hold.

3. Suppose that $\mathcal{U}'(R_s)(x', y') \wedge r(x, x')$ holds. Let $y = \mathbf{U}(x, R_s)$. By Lemma 7.14, $r(y, y')$ holds. Hence $\mathcal{U}(R_s)(x, y) \wedge r(y, y')$ holds.
4. By the definition of r and the assertion (38) of Lemma 7.14, if $r(x, x')$ holds then $x \in \mathcal{C}(A)$ implies $x' \in \mathcal{C}'(A)$. ◁

To increase readability we recall Theorem 7.9 before presenting its proof.

Theorem 7.9. *Let $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ be a deterministic Horn knowledge base in $\mathcal{R}eg$, and $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ the pseudo-interpretation constructed by Algorithm 1 for $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ w.r.t. $\mathfrak{s} \in \{\mathsf{Sem}_1, \mathsf{Sem}_{2,\mathcal{R}}\}$. Then $\mathcal{I}$ is a least $\mathfrak{s}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.*

Proof. By Lemma 7.13, $\mathcal{I}$ is an $\mathfrak{s}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. By Lemmas 7.15 and 6.8, for every $\mathfrak{s}$-pseudo-model $\mathcal{I}'$ of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, we have that $\mathcal{I} \leq_{\mathfrak{s}} \mathcal{I}'$. ◁

8 Characterizations of Least Pseudo-Models

In this section, if not stated otherwise, by a *formula* we mean a formula possibly with $\forall$ and $\forall\exists$ but without automaton-modal operators. We will prove the following theorem and present its consequences.

Theorem 8.1. *Let $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ be a deterministic Horn knowledge base in $\mathcal{R}eg$, $\mathcal{I}$ a least $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$, C a positive allsome-formula, and a an individual. Then $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$ iff $\mathcal{I} \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$.*

As mentioned earlier, we do not have a similar result for Sem_1.

Definition 8.2. Let $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ be a deterministic Horn knowledge base in $\mathcal{R}eg$, $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ be the least $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ constructed by Algorithm 1, and $\mathcal{E}_0$ and $\mathbf{U}$ be the mappings used for defining $\mathcal{U}$. For every role name R_t, define

$$\mathcal{E}'_0(R_t) = \mathcal{U}'_0(R_t) = \mathcal{E}_0(R_t) \cup \{(x, y) \mid \mathbf{U}(x, R_t) = y \text{ and } \exists y'.(\mathcal{E}(R_t)(x, y'))\}. \quad (42)$$

Let $\mathcal{E}'$ (resp. $\mathcal{U}'$) be the least extension of $\mathcal{E}'_0$ (resp. $\mathcal{U}'_0$) that satisfies the role axioms of $\mathcal{R}$. Then $\mathcal{I}' = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}', \mathcal{U}' \rangle$ can be treated as an interpretation since $\mathcal{E}' = \mathcal{U}'$. We call it the *interpretation corresponding to* $\mathcal{I}$. ◁

Notice the occurrence of $\mathcal{E}$ in (42). Also note that

$$\mathcal{E}_0(R_t) \subseteq \mathcal{E}'_0(R_t) = \mathcal{U}'_0(R_t) \subseteq \mathcal{U}_0(R_t).$$

To prove Theorem 8.1 we need the following lemma.

Lemma 8.3. *Let $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ be a deterministic Horn knowledge base in $\mathcal{R}eg$, $\mathcal{I}$ the least $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ constructed by Algorithm 1, $\mathcal{I}'$ the interpretation corresponding to $\mathcal{I}$, and C a positive allsome-formula. Let*

$$r = \{(a, a) \mid a \text{ is an individual occurring in } \mathcal{A}\} \cup \{(x, x') \in \Delta \times \Delta \mid x \text{ is not an individual and } \mathcal{I}, x' \models_{\mathsf{Sem}_{2,\mathcal{R}}} \mathbf{C}(x)\},$$

where Δ and $\mathbf{C}$ are the data structures used in the execution of Algorithm 1. Then for every $x, x' \in \Delta$, if $r(x, x')$ holds and $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$ then $\mathcal{I}, x' \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$. Moreover, for every $x \in \Delta$, $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$ implies $\mathcal{I}, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$. As a consequence, if $\mathcal{I}'$ validates C w.r.t. $\mathsf{Sem}_{2,\mathcal{R}}$ then $\mathcal{I}$ also validates C w.r.t. $\mathsf{Sem}_{2,\mathcal{R}}$.

Proof. We will refer to the data structures used in the execution of Algorithm 1. By Lemma 7.13, $\mathcal{I}, y \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$ for every $y \in \Delta$ and $D \in \mathbf{C}(y)$. Hence, r is reflexive, and it suffices to prove the first assertion of the lemma. By Lemma 7.15, $\mathcal{I} \leq_r \mathcal{I}$. Furthermore, by Lemma 7.14, we have that:

$$\forall u, v, u', v', R_s \;\; r(u, u') \wedge (\mathbf{U}(u, R_s) = v) \wedge \mathcal{U}(R_s)(u', v') \rightarrow r(v, v'). \tag{43}$$

Let $\mathcal{I}' = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}', \mathcal{U}' \rangle$. We prove the assertion by induction on the construction of C. Suppose that $r(x, x')$ holds and $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$. We show that $\mathcal{I}, x' \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$.

- Case $C = A$: Since $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} A$, we have that $A \in \mathbf{C}(x)$. Since $\mathcal{I} \leq_r \mathcal{I}$ and $r(x, x')$ holds, it follows that $A \in \mathbf{C}(x')$. Hence $\mathcal{I}, x' \models_{\mathsf{Sem}_{2,\mathcal{R}}} A$.
- Case $C = D \sqcap D'$ or $C = D \sqcup D'$ is trivial.
- Case $C = \exists R_t.D$: Since $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$, there exists y such that $\mathcal{E}'(R_t)(x, y)$ holds and $\mathcal{I}', y \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$. Consequently, there are $x_0, \ldots, x_k$ of Δ such that $x_0 = x$, $x_k = y$, $\mathcal{E}'_0(R_{s_i})(x_{i-1}, x_i)$ holds for $1 \leq i \leq k$ and $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ is a consequence of $\mathcal{R}$. Let $x'_0 = x'$. For $1 \leq i \leq k$, choose x'_i as follows:
  - Case $\mathcal{E}_0(R_{s_i})(x_{i-1}, x_i)$ holds: Since $\mathcal{I} \leq_r \mathcal{I}$ and $r(x_{i-1}, x'_{i-1})$ holds, there exists $x'_i \in \Delta$ such that $r(x_i, x'_i)$ and $\mathcal{E}(R_{s_i})(x'_{i-1}, x'_i)$ hold.
  - Case $x_i = \mathbf{U}(x_{i-1}, R_{s_i})$: There must exist z_i such that $\mathcal{E}(R_{s_i})(x_{i-1}, z_i)$ holds. Analogously as for the above case, there exists $x'_i \in \Delta$ such that $r(z_i, x'_i)$ and $\mathcal{E}(R_{s_i})(x'_{i-1}, x'_i)$ hold. By (43), it follows that $r(x_i, x'_i)$ holds.

  Since $r(x_k, x'_k)$ holds and $y = x_k$ and $\mathcal{I}', y \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$, by the inductive assumption, $\mathcal{I}, x'_k \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$. Since $\mathcal{E}(R_{s_i})(x'_{i-1}, x'_i)$ holds for $1 \leq i \leq k$ and $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ is a consequence of $\mathcal{R}$, $\mathcal{E}(R_t)(x'_0, x'_k)$ must hold. It follows that $\mathcal{I}, x'_0 \models_{\mathsf{Sem}_{2,\mathcal{R}}} \exists R_t.D$, which means $\mathcal{I}, x' \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$.
- Case $C = \forall \exists R_t.D$: Since $\mathcal{I} \leq_r \mathcal{I}$ and $r(x, x')$ holds, by Lemma 6.8, it suffices to show that $\mathcal{I}, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$. Let $x_0 \mathcal{U}_0(R_{s_1}) x_1 \ldots \mathcal{U}_0(R_{s_k}) x_k$ be a path such that $x_0 = x$ and $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ is a consequence of $\mathcal{R}$. It suffices to show that $\mathcal{I}, x_k \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$, and for every $1 \leq i \leq k$, $\mathcal{E}(R_{s_i})(x_{i-1}, y_i)$ holds for some y_i.

  We prove by an inner induction on $1 \leq i \leq k$ that $\mathcal{U}'_0(R_{s_i})(x_{i-1}, x_i)$ holds. Fix $1 \leq i \leq k$ and inductively assume that $\mathcal{U}'_0(R_{s_j})(x_{j-1}, x_j)$ holds for every $1 \leq j \leq i-1$. Since $\mathcal{I}', x_0 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$, it follows that $\mathcal{E}'(R_{s_i})(x_{i-1}, z_i)$ holds for some z_i. Hence, there are $u_0, \ldots, u_h \in \Delta$ and $q_1, \ldots, q_h \in \mathrm{IND}$ (which are dependent on i) such that $u_0 = x_{i-1}$, $u_h = z_i$, $\mathcal{E}'_0(R_{q_l})(u_{l-1}, u_l)$ holds for $1 \leq l \leq h$ and $R_{q_1} \circ \ldots \circ R_{q_h} \sqsubseteq R_{s_i}$ is a consequence of $\mathcal{R}$.

  We prove by a second inner induction on l from h down to 1 that $\mathcal{I}, u_{l-1} \models_{\mathsf{Sem}_{2,\mathcal{R}}} \exists R_{q_l} \ldots . \exists R_{q_h}.\top$. Since $\mathcal{E}'_0(R_{q_l})(u_{l-1}, u_l)$ holds, either $\mathcal{E}_0(R_{q_l})(u_{l-1}, u_l)$ holds or $u_l = \mathbf{U}(u_{l-1}, R_{q_l})$ and $\mathcal{E}(R_{q_l})(u_{l-1}, u'_l)$ holds for some u'_l. As the assumption of the second inner induction, we have that $\mathcal{I}, u_l \models_{\mathsf{Sem}_{2,\mathcal{R}}} \exists R_{q_{l+1}} \ldots . \exists R_{q_h}.\top$. If $\mathcal{E}_0(R_{q_l})(u_{l-1}, u_l)$ holds, then it follows that $\mathcal{I}, u_{l-1} \models_{\mathsf{Sem}_{2,\mathcal{R}}} \exists R_{q_l} \ldots . \exists R_{q_h}.\top$. Otherwise, since $r(u_{l-1}, u_{l-1})$ holds and by (43), $r(u_l, u'_l)$ holds, which implies that $\mathcal{I}, u'_l \models_{\mathsf{Sem}_{2,\mathcal{R}}}$

$\exists R_{q_{l+1}}.\ldots.\exists R_{q_h}.\top$ (by Lemma 6.8), and hence $\mathcal{I}, u_{l-1} \models_{\mathsf{Sem}_{2,\mathcal{R}}} \exists R_{q_l}.\ldots.\exists R_{q_h}.\top$. Thereby we have proved the second inner induction, which implies that $\mathcal{I}, u_0 \models_{\mathsf{Sem}_{2,\mathcal{R}}} \exists R_{q_1}.\ldots.\exists R_{q_h}.\top$. Consequently, $\mathcal{I}, x_{i-1} \models_{\mathsf{Sem}_{2,\mathcal{R}}} \exists R_{s_i}.\top$, since $x_{i-1} = u_0$ and $R_{q_1} \circ \ldots \circ R_{q_h} \sqsubseteq R_{s_i}$ is a consequence of $\mathcal{R}$. It follows that $\mathcal{E}(R_{s_i})(x_{i-1}, y_i)$ holds for some y_i (one of the main goals). This together with $\mathcal{U}_0(R_{s_i})(x_{i-1}, x_i)$ implies that $\mathcal{U}'_0(R_{s_i})(x_{i-1}, x_i)$ holds. Thereby we have proved the first inner induction. Since $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$, i.e. $\mathcal{I}', x_0 \models_{\mathsf{Sem}_{2,\mathcal{R}}} \forall\exists R_t.D$, and $R_{s_1} \circ \ldots \circ R_{s_k} \sqsubseteq R_t$ is a consequence of $\mathcal{R}$, it follows that $\mathcal{I}', x_k \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$. Since r is reflexive, by the outer inductive assumption, $\mathcal{I}, x_k \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$. This completes the proof. ◁

Proof (of Theorem 8.1). Suppose that $\mathcal{I} \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$. Let $\mathcal{I}''$ be an arbitrary $\mathsf{Sem}_{2,\mathcal{R}}$-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. It can be treated as a $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. Since $\mathcal{I}$ is a least $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ and $\mathcal{I} \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$, we have that $\mathcal{I}'' \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$. This implies that $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$.

For the converse, suppose that $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$. Without loss of generality, we assume that $\mathcal{I} = \langle \Delta, \mathcal{O}, \mathcal{C}, \mathcal{E}, \mathcal{U} \rangle$ is the least $\mathsf{Sem}_{2,\mathcal{R}}$-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ constructed by Algorithm 1. We will refer to the data structures used in the execution of Algorithm 1. Let $\mathcal{I}'$ be the interpretation corresponding to $\mathcal{I}$. We show that $\mathcal{I}'$ is a $\mathsf{Sem}_{2,\mathcal{R}}$-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. This will imply that $\mathcal{I}' \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$, and hence $\mathcal{I} \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$, by Lemma 8.3. Clearly, $\mathcal{I}'$ is a model of $\mathcal{R}$ and $\mathcal{A}$. To prove that $\mathcal{I}'$ is a $\mathsf{Sem}_{2,\mathcal{R}}$-model of $\mathcal{T}$, it suffices to prove by induction on the construction of C that for every $x \in \Delta$ and $C \in \mathbf{C}(x)$ without automaton-modal operators, $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$.

The case when C is an atomic concept is trivial.

Consider the case when C is of the form $D \sqcap D'$. By Step 2a of Algorithm 1, $\{D, D'\} \subset \mathbf{C}(x)$ (note that if $x \notin \Delta_0$ then the last execution of Step 2(a)ii gives $x_* = x$, because otherwise x would become unreachable from Δ_0 and then be deleted by Step 5). By the inductive assumption, it follows that $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$ and $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} D'$. Hence $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} D \sqcap D'$, i.e. $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$.

Consider the case when C is of the form $D \sqsubseteq D'$. Suppose that $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$. By Lemma 8.3, $\mathcal{I}, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$. Since $\mathbf{U}(u, R_t)$ is defined for every $u \in \Delta$ and every role name R_t, we also have that $\mathcal{I}, x \models_{\mathfrak{s}c} D$ with $\mathfrak{s} = \mathsf{Sem}_{2,\mathcal{R}}$. By Step 2b of Algorithm 1, $D' \in \mathbf{C}(x)$ (note that if $x \notin \Delta_0$ then the last execution of Step 2(b)ii gives $x_* = x$). By the inductive assumption, it follows that $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} D'$. Hence $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$.

Consider the case when C is of the form $\forall R_t.D$. Let y be an arbitrary element of Δ such that $\mathcal{U}'(R_t)(x, y)$ holds. Since $\mathcal{U}'(R_t) \subseteq \mathcal{U}(R_t)$, it can be seen that $D \in \mathbf{C}(y)$. By the inductive assumption, $\mathcal{I}', y \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$. Hence $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$.

Consider the case when C is of the form $\exists R_t.D$. Let $y = \mathbf{E}(x, \exists R_t.D)$. Thus $D \in \mathbf{C}(y)$. We also have that $\mathcal{E}'(R_t)(x, y)$ holds. By the inductive assumption, $\mathcal{I}', y \models_{\mathsf{Sem}_{2,\mathcal{R}}} D$. Hence $\mathcal{I}', x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$. This completes the proof. ◁

As a consequence of Theorem 8.1, we have:

Theorem 8.4. *Let $(\mathcal{R}, \mathcal{T}', \mathcal{A})$, $\mathcal{T}$, $\mathcal{I}_1$, $\mathcal{I}_2$, C, C', and a be as in Lemma 6.10, i.e. :*

- *$(\mathcal{R}, \mathcal{T}', \mathcal{A})$ is a Horn knowledge base in $\mathcal{R}eg$,*
- *$\mathcal{T}$ is the deterministic version of $\mathcal{T}'$,*
- *$\mathcal{I}_1$ is a least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$,*

- $\mathcal{I}_2$ *is a least* $\mathsf{Sem}_{2,\mathcal{R}}$*-pseudo-model of* $(\mathcal{R}, \mathcal{T}, \mathcal{A})$,
- C *is a positive formula (possibly with* $\forall$ *and* $\forall\exists$*),*
- C' *is the formula obtained from* C *by replacing every subformula of the form* $\forall\exists R_t.D$ *by* $(\forall R_t.D \sqcap \exists R_t.D)$,
- a *is an individual.*

Suppose that, for every $t \in \mathrm{IND}$,

1. *either the constructor* $\forall R_t.D$ *does not occur in* C' *and the premises of the program clauses of* $\mathcal{T}'$
2. *or* $\exists R_s.\top \in \mathcal{T}'$ *for every* $s \in \mathrm{IND}$ *occurring in some word accepted by* $\mathbf{A}_t$
3. *or* $\mathcal{L}(\mathbf{A}_t) = \{t\}$ *and the constructor* $\forall R_t.D$ *may occur in* C' *and the premises of the program clauses of* $\mathcal{T}'$ *only in the form* $(\forall R_t.D \sqcap \exists R_t.D')$ *for some arbitrary* D'.

Then the three assertions $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C'(a)$, $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$ *and* $\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$ *are equivalent.*

Proof. By Lemma 6.10, we only need to show that $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C'(a)$ implies $\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$. Let IND_3 be the set of indices t satisfying Condition 3. Let $\mathcal{I}$ be an arbitrary model of (the TBox) $\{\exists R_s.\top \in \mathcal{T}' \mid s \in \mathrm{IND}\}$.

Let E and E' be formulas such that:

- either E' occurs in a premise of a program of $\mathcal{T}'$ and E is obtained from E' by replacing every occurrence of $\forall$ by $\forall\exists$
- or E is C or a subformula of C and E' is obtained from E by replacing every subformula of the form $\forall\exists R_t.D$ by $(\forall R_t.D \sqcap \exists R_t.D)$;
- and furthermore, if $t \in \mathrm{IND}_3$ then a formula of the form $\forall R_t.D$ may occur in E' only in the form $(\forall R_t.D \sqcap \exists R_t.D')$ for some arbitrary D'.

It is straightforward to prove by induction on the structures of E and E' that, for every $x \in \Delta^{\mathcal{I}}$, we have that $\mathcal{I}, x \models E'$ iff $\mathcal{I}, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} E$. As a consequence, $\mathcal{I} \models (\mathcal{R}, \mathcal{T}', \mathcal{A})$ iff $\mathcal{I} \models_{\mathsf{Sem}_{2,\mathcal{R}}} (\mathcal{R}, \mathcal{T}, \mathcal{A})$, and $\mathcal{I} \models C'(a)$ iff $\mathcal{I} \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$. Hence,

$$(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C'(a) \;\text{ iff }\; (\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a). \tag{44}$$

Note that if $t \in \mathrm{IND}_3$ then the constructor $\forall R_t.D$ may occur in C only in the form $(\forall R_t.D \sqcap \exists R_t.D')$ for some arbitrary D'. Let $C_{\forall\exists}$ be the formula obtained from C by replacing every subformula $\forall R_t.D$ by $\forall\exists R_t.D$. Similarly as shown above:

- for every $x \in \Delta^{\mathcal{I}}$, we have that $\mathcal{I}, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$ iff $\mathcal{I}, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C_{\forall\exists}$
- for every $x \in \Delta^{\mathcal{I}_2}$, we have that $\mathcal{I}_2, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C$ iff $\mathcal{I}_2, x \models_{\mathsf{Sem}_{2,\mathcal{R}}} C_{\forall\exists}$.

Therefore, $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$ iff $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models_{\mathsf{Sem}_{2,\mathcal{R}}} C_{\forall\exists}(a)$, and by Theorem 8.1, iff $\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C_{\forall\exists}(a)$, and hence, iff $\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$. By (44), it follows that $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C'(a)$ iff $\mathcal{I}_2 \models_{\mathsf{Sem}_{2,\mathcal{R}}} C(a)$, which completes the proof.

Corollary 8.5. *Let* $(\mathcal{R}, \mathcal{T}', \mathcal{A})$ *be a Horn knowledge base in* $\mathcal{R}eg$ *and* C *be a positive formula without the constructor* $\forall\exists$ *such that, for every* $t \in \mathrm{IND}$,

- *either the constructor* $\forall R_t.D$ *does not occur in* C *and the premises of the program clauses of* $\mathcal{T}'$

- *or* $\exists R_s.\top \in \mathcal{T}'$ *for every* $s \in \mathrm{IND}$ *occurring in some word accepted by* $\mathbf{A}_t$
- *or* $\mathcal{L}(\mathbf{A}_t) = \{t\}$ *and the constructor* $\forall R_t.D$ *may occur in* C *and the premises of the program clauses of* $\mathcal{T}'$ *only in the form* $(\forall R_t.D \sqcap \exists R_t.D')$ *for some arbitrary* D'.

Then

1. *for* $\mathcal{T}$ *being the deterministic version of* $\mathcal{T}'$ *and* $\mathcal{I}_1$ *being a least* Sem_1*-pseudo-model of* $(\mathcal{R}, \mathcal{T}, \mathcal{A})$*, the approximation of checking* $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C(a)$ *by checking* $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$ *is exact*
2. *checking* $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C(a)$ *can be done in polynomial time in the size of* $\mathcal{A}$.

Proof. The first assertion immediately follows from Theorem 8.4 (as $C' = C$). For the second assertion, note that checking whether $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C(a)$ can be done by constructing $\mathcal{I}_1$ and checking whether $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$. By Proposition 7.8, $\mathcal{I}_1$ can be constructed in polynomial time in the size of $\mathcal{A}$. Checking whether $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$ can be done in polynomial time in the size of $\mathcal{I}_1$ (by Proposition 6.9), and hence in polynomial time in the size of $\mathcal{A}$.

9 Conclusions

Since instance checking is a basic task of knowledge bases in DLs, developing a good formalism and an efficient decision procedure for the instance checking problem is desirable for practical applications. The data complexity of this problem is coNP-hard even for Horn knowledge bases in $\mathcal{ALC}$ (with $\mathcal{R} = \emptyset$). In this paper, we have studied weakenings with PTime data complexity of the instance checking problem for Horn knowledge bases in $\mathcal{R}eg$. We have established important cases when the weakenings give an exact approximation.

Given a Horn knowledge base $(\mathcal{R}, \mathcal{T}', \mathcal{A})$ in $\mathcal{R}eg$ and a positive formula C without the constructor $\forall\exists$, we propose to approximate the checking whether $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C(a)$ by checking whether $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$, where $\mathcal{I}_1$ is the least Sem_1-pseudo-model of $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ constructed by Algorithm 1 for $\mathcal{T}$ being the deterministic version of $\mathcal{T}'$. This is a weakening in the sense that $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$ implies $(\mathcal{R}, \mathcal{T}', \mathcal{A}) \models C(a)$. The pseudo-model $\mathcal{I}_1$ is constructed in polynomial time in the size of $\mathcal{A}$, and checking $\mathcal{I}_1 \models_{\mathsf{Sem}_1} C(a)$ can be done in polynomial time in the size of $\mathcal{I}_1$ and C. That is, the weakening has PTime data complexity.

The approximation is exact for the cases mentioned in Corollary 8.5. The examples given in Section 5 show that the assumptions of these cases are acceptable for certain domains of application. Our results stated in Corollary 8.5 are the strongest ones that are known about Horn fragments with PTime data complexity of the description logics $\mathcal{ALC}$ and $\mathcal{R}eg$. Recall that all of DHL [16], Horn-$\mathcal{SHIQ}$ [20], *DL-Lite* [5] and $\mathcal{EL}$ [22, 24, 37] disallow the constructor $\forall R.C$ (and do not use $\forall\exists R.C$) in premises of program clauses and goals. Also recall that our notion of regular RBox of $\mathcal{R}eg$ is strictly more general than the notion of RBox of transitive roles and role hierarchies used in [16, 20, 5, 37] and the notion of acyclic generalized RBox used in [24] (and called there "regular RBox").

Our method is therefore very promising for practical applications of DLs. It has been developed using bottom-up computation and to obtain a good theoretical result

(PTime data complexity). For efficiency, optimizations are expected, e.g. by combining the method with the top-down (goal-directed) approach to reduce the search space. We briefly discuss some optimizations in Appendix B. As future work, we will extend our method to cope also with inverse roles and number restrictions.

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In L.P. Kaelbling and A. Saffiotti, editors, *Proceedings of IJCAI'2005*, pages 364–369. Morgan-Kaufmann Publishers, 2005.
2. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
3. M. Baldoni, L. Giordano, and A. Martelli. A tableau for multimodal logics and some (un)decidability results. In H.C.M. de Swart, editor, *Proceedings of TABLEAUX'1998, LNCS 1397*, pages 44–59, 1998.
4. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else? In R.L. de Mántaras and L. Saitta, editors, *Proceedings of ECAI'2004*, pages 298–302. IOS Press, 2004.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The L-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
6. D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 1581–1634. Elsevier, 2001.
7. F. Debart, P. Enjalbert, and M. Lescot. Multimodal logic programming using equational and order-sorted logic. *Theoretical Comp. Science*, 105:141–166, 1992.
8. S. Demri. The complexity of regularity in grammar logics and related modal logics. *Journal of Logic and Computation*, 11(6):933–960, 2001.
9. S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. *Journal of Logic, Language and Information*, 14(3):289–329, 2005.
10. P. Doherty, W. Łukaszewicz, and A. Szałas. Communication between agents with heterogeneous perceptual capabilities. *Journal of Information Fusion*, 8(1):56–69, 2007.
11. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. Fusing approximate knowledge from distributed sources. In G.A. Papadopoulos and C. Badica, editors, *Proceedings of IDC'2009*, volume 237 of *Studies in Computational Intelligence*, pages 75–86. Springer, 2009.
12. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. Tractable approximate knowledge fusion using the Horn fragment of serial propositional dynamic logic. *Int. J. Approx. Reasoning*, 51(3):346–362, 2010.
13. M.J. Fischer and R.E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.
14. R. Goré. Tableau methods for modal and temporal logics. In D'Agostino, Gabbay, Hähnle, and Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, 1999.
15. R. Goré and L.A. Nguyen. A tableau system with automaton-labelled formulae for regular grammar logics. In B. Beckert, editor, *Proceedings of TABLEAUX 2005, LNAI 3702*, pages 138–152. Springer-Verlag, 2005.
16. B.N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. In *WWW*, pages 48–57, 2003.
17. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
18. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *Proceedings of KR'2006*, pages 57–67. AAAI Press, 2006.
19. I. Horrocks and U. Sattler. Decidability of SHIQ with complex role inclusion axioms. *Artificial Intelligence*, 160(1-2):79–104, 2004.
20. U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive Datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.
21. K. Konolige. Belief and incompleteness. Technical Report 319, SRI Inter., 1984.
22. A. Krisnadhi and C. Lutz. Data complexity in the *el* family of description logics. In N. Dershowitz and A. Voronkov, editors, *Proceedings of LPAR'2007, LNCS 4790*, pages 333–347. Springer, 2007.

23. M. Krötzsch, S. Rudolph, and P. Hitzler. Complexity boundaries for Horn description logics. In *Proceedings of AAAI'2007*, pages 452–457. AAAI Press, 2007.
24. M. Krötzsch, S. Rudolph, and P. Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In *Proceedings of ISWC'2007 + ASWC'2007, LNCS 4825*, pages 310–323. Springer, 2007.
25. A. Mateescu and A. Salomaa. Formal languages: an introduction and a synopsis. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages - Volume 1: Word, Language, Grammar*, pages 1–40. Springer, 1997.
26. J. McCarthy. First order theories of individual concepts and propositions. *Machine Intelligence*, 9:120–147, 1979.
27. L.A. Nguyen. Constructing the least models for positive modal logic programs. *Fundamenta Informaticae*, 42(1):29–60, 2000.
28. L.A. Nguyen. On the complexity of fragments of modal logics. In R.A. Schmidt et al., editor, *Advances in Modal Logic - Volume 5*, pages 249–268. King's College Publications, 2004.
29. L.A. Nguyen. A bottom-up method for the deterministic Horn fragment of the description logic $\mathcal{ALC}$. In M. Fisher et al., editor, *Proceedings of JELIA 2006, LNAI 4160*, pages 346–358. Springer-Verlag, 2006.
30. L.A. Nguyen. Multimodal logic programming. *Theoretical Computer Science*, 360:247–288, 2006.
31. L.A. Nguyen. On the deterministic Horn fragment of test-free PDL. In I. Hodkinson and Y. Venema, editors, *Advances in Modal Logic - Volume 6*, pages 373–392. King's College Publications, 2006.
32. L.A. Nguyen. Approximating Horn knowledge bases in regular description logics to have PTIME data complexity. In V. Dahl and I. Niemelä, editors, *Proceedings of ICLP'2007, LNCS 4670*, pages 438–439. Springer, 2007.
33. L.A. Nguyen. Weakening Horn knowledge bases in regular description logics to have PTIME data complexity. In S. Ghilardi, U. Sattler, V. Sofronie-Stokkermans, and A. Tiwari, editors, *Proceedings of Automated Deduction: Decidability, Complexity, Tractability ADDCT'07*, pages 32–47, 2007.
34. L.A. Nguyen. Constructing finite least Kripke models for positive logic programs in serial regular grammar logics. *Logic Journal of the IGPL*, 16(2):175–193, 2008.
35. A. Nonnengart. How to use modalities and sorts in Prolog. In C. MacNish, D. Pearce, and L.M. Pereira, editors, *Proceedings of JELIA'94, LNCS 838*, pages 365–378. Springer, 1994.
36. V.R. Pratt. Models of program logics. In *Proceedings of FOCS'1979*, pages 115–122. IEEE, 1979.
37. R. Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proceedings of DL'2007*, pages 451–458.
38. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.
39. K. Schild. Terminological cycles and the propositional μ-calculus. In *Proceedings of KR'1994*, pages 509–520, 1994.

A Hardness of the General Horn Fragment of $\mathcal{ALC}$

Theorem A.1. *The data complexity of the instance checking problem for the general Horn fragment of $\mathcal{ALC}$ is coNP-complete. That is, given $\mathcal{R} = \emptyset$, a positive logic program $\mathcal{T}$, an ABox $\mathcal{A}$, a positive formula C without the constructor $\forall\exists$, and an individual a, checking whether $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$ is coNP-complete w.r.t. the size of $\mathcal{A}$.*

Proof. The upper bound (coNP) follows from the result of Hustadt et al. [20] that the data complexity of the instance checking problem in $\mathcal{SHIQ}$ is coNP-complete. For the lower bound, we show that the checking problem $(\emptyset, \mathcal{T}, \mathcal{A}) \not\models C(a)$ is NP-hard w.r.t. the size of $\mathcal{A}$ by using a reduction from the 3SAT problem, which is known to be NP-hard. The technique used for this comes from our proof of that checking satisfiability of a set of Horn formulas with modal depth bounded by 2 in the modal logic K is NP-complete [28].

The 3SAT problem is to check satisfiability of a clause set $X = \{C_1, \ldots, C_m\}$, where $C_i = L1_i \vee L2_i \vee L3_i$ and $L1_i$, $L2_i$, $L3_i$ are classical literals. Given such a set X, we construct in polynomial time a positive logic program $\mathcal{T}$ and an ABox $\mathcal{A}$ such that X is satisfiable iff $(\emptyset, \mathcal{T}, \mathcal{A}) \not\models \mathit{inconsistent}(\tau)$, where *inconsistent* is an atomic concept and τ is an individual.

Let $a_1, \ldots, a_l$ be all the primitive propositions occurring in the clauses of X. We use the same name a_i, for $1 \le i \le l$, to denote an individual (corresponding to the primitive proposition a_i). We use also individuals $c_1, \ldots, c_m$ (where c_i corresponds to the clause C_i) and τ (as the *actual world* in a Kripke model).

We use atomic concepts C, $L1$, $L2$, $L3$, T, F, *tau*, *inconsistent*, which have the following intuition:

- $C(x)$ means that $x = c_i$ for some $1 \le i \le m$ and corresponds to *clause* C_i.
- $L1(x)$ means that $x = c_i$ for some $1 \le i \le m$ and the *1st literal* of C_i has value *true*.
- $L2(x)$ and $L3(x)$ have a similar meaning as $L1(x)$.
- $T(x)$ (resp. $F(x)$) means that x is one of $a_1, \ldots, a_l$ and its corresponding primitive proposition has value *true* (resp. *false*).
- $\mathit{tau}(x)$ means $x = \tau$,
- *inconsistent* is interpreted by the empty set (like $\neg\top$).

Let $\mathrm{IND} = \{0, 1, 2, p_1, p_2, p_3, n_1, n_2, n_3\}$ and

$$\begin{aligned}\mathcal{A} = \{&C(c_1), \ldots, C(c_m), \mathit{tau}(\tau)\} \cup \{R_0(\tau, a_i) \mid 1 \le i \le l\} \cup \\ &\{R_{p_i}(c_j, a_k) \mid 1 \le i \le 3, 1 \le j \le m, 1 \le k \le l \text{ and } a_k \text{ is the } i\text{th literal of } C_j\} \cup \\ &\{R_{n_i}(c_j, a_k) \mid 1 \le i \le 3, 1 \le j \le m, 1 \le k \le l \text{ and } \neg a_k \text{ is the } i\text{th literal of } C_j\}.\end{aligned}$$

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

$$C \sqcap \exists R_1.\top \sqsubseteq L1 \tag{45}$$
$$C \sqcap \exists R_2.\top \sqsubseteq L2 \tag{46}$$
$$C \sqcap \forall R_1.\mathit{inconsistent} \sqcap \forall R_2.\mathit{inconsistent} \sqsubseteq L3 \tag{47}$$
$$L1 \sqsubseteq \forall R_{p_1}.T \sqcap \forall R_{n_1}.F \tag{48}$$
$$L2 \sqsubseteq \forall R_{p_2}.T \sqcap \forall R_{n_2}.F \tag{49}$$
$$L3 \sqsubseteq \forall R_{p_3}.T \sqcap \forall R_{n_3}.F \tag{50}$$
$$\mathit{tau} \sqcap \exists R_0.(T \sqcap F) \sqsubseteq \mathit{inconsistent} \tag{51}$$

We show that X is satisfiable iff $(\emptyset, \mathcal{T}, \mathcal{A}) \not\models \mathit{inconsistent}(\tau)$.

Suppose that X is satisfied by an assignment V. We build a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ such that $\mathcal{I} \not\models \mathit{inconsistent}(\tau)$. Let $\Delta^{\mathcal{I}}$ consist of all the mentioned individuals plus ω. For x being one of the individuals, let $x^{\mathcal{I}} = x$. Let $\mathit{inconsistent}^{\mathcal{I}} = \emptyset$ and let the interpretation of C, R_0, R_{p_1}, R_{p_2}, R_{p_3}, R_{n_1}, R_{n_2} and R_{n_3} in $\mathcal{I}$ be specified by the ABox $\mathcal{A}$ (e.g. $C^{\mathcal{I}} = \{c_1, \ldots, c_m\}$). Let the interpretation of the other atomic concepts and roles be specified as follows:

$$
\begin{array}{l}
T^{\mathcal{I}} = \{a_i \mid 1 \leq i \leq l, V(a_i) = \mathit{true}\} \\
F^{\mathcal{I}} = \{a_i \mid 1 \leq i \leq l, V(a_i) = \mathit{false}\} \\
\\
L1^{\mathcal{I}} = \{c_i \mid 1 \leq i \leq m, \text{the 1st literal of } C_i \text{ has value } \mathit{true} \text{ by using } V\} \\
L2^{\mathcal{I}} = \{c_i \mid 1 \leq i \leq m, \text{the 2nd literal of } C_i \text{ has value } \mathit{true} \text{ by using } V\} \\
L3^{\mathcal{I}} = \{c_i \mid 1 \leq i \leq m, \text{the 3rd literal of } C_i \text{ has value } \mathit{true} \text{ by using } V\} \\
\\
R_1^{\mathcal{I}} = \{(c_i, \omega) \mid 1 \leq i \leq m, c_i \in L1^{\mathcal{I}}\} \\
R_2^{\mathcal{I}} = \{(c_i, \omega) \mid 1 \leq i \leq m, c_i \in L2^{\mathcal{I}}\}
\end{array}
$$

We need to show that $\mathcal{I}$ is a model of $\mathcal{T}$.

$\mathcal{I}$ validates program clauses (45) and (46) of $\mathcal{T}$ due to the definition of $R_1^{\mathcal{I}}$ and $R_2^{\mathcal{I}}$.

Consider program clause (47) of $\mathcal{T}$. Suppose that $x \in C^{\mathcal{I}}$ and $x \notin L3^{\mathcal{I}}$. We show that $x \notin (\forall R_1.\mathit{inconsistent} \sqcap \forall R_2.\mathit{inconsistent})^{\mathcal{I}}$. We have that $x = c_i$ for some $1 \leq i \leq m$ and the 3rd literal of C_i has value false by using V. Thus, either the 1st literal or the 2nd literal of C_i has value true by using V. It follows that either $c_i \in L1^{\mathcal{I}}$ or $c_i \in L2^{\mathcal{I}}$. Consequently, either $R_1^{\mathcal{I}}(c_i, \omega)$ or $R_2^{\mathcal{I}}(c_i, \omega)$ holds. Hence $x \notin (\forall R_1.\mathit{inconsistent} \sqcap \forall R_2.\mathit{inconsistent})^{\mathcal{I}}$ since $\mathit{inconsistent}^{\mathcal{I}} = \emptyset$.

It is easy to check that $\mathcal{I}$ validates program clauses (48), (49), (50) of $\mathcal{T}$.

Since $T^{\mathcal{I}} \cap F^{\mathcal{I}} = \emptyset$, $\mathcal{I}$ validates program clause (51) of $\mathcal{T}$.

For the converse, suppose that $(\emptyset, \mathcal{T}, \mathcal{A}) \not\models \mathit{inconsistent}(\tau)$, i.e. there exists a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ such that $\tau^{\mathcal{I}} \notin \mathit{inconsistent}^{\mathcal{I}}$. We show that X is satisfiable.

As $\mathcal{I}$ validates program clause (51) of $\mathcal{T}$ but $\tau^{\mathcal{I}} \notin \mathit{inconsistent}^{\mathcal{I}}$, no $a_i^{\mathcal{I}}$ belongs to both $T^{\mathcal{I}}$ and $F^{\mathcal{I}}$. Let V be the assignment such that, for $1 \leq i \leq l$, $V(a_i) = \mathit{true}$ iff $a_i^{\mathcal{I}} \in T^{\mathcal{I}}$.

Since $\mathcal{I}$ validates program clauses (45), (46), (47) of $\mathcal{T}$, for every $1 \leq i \leq m$, either $c_i^{\mathcal{I}} \in L1^{\mathcal{I}}$ or $c_i^{\mathcal{I}} \in L2^{\mathcal{I}}$ or $c_i^{\mathcal{I}} \in L3^{\mathcal{I}}$. Suppose that $c_i^{\mathcal{I}} \in L1^{\mathcal{I}}$. We show that the 1st literal of C_i has value true by using V. Since $\mathcal{I}$ validates program clause (48) of $\mathcal{T}$, we have that $c_i^{\mathcal{I}} \in (\forall R_{p_1}.T)^{\mathcal{I}}$ and $c_i^{\mathcal{I}} \in (\forall R_{n_1}.F)^{\mathcal{I}}$. If the 1st literal of C_i is a_k for some $1 \leq k \leq l$, then $a_k^{\mathcal{I}} \in T^{\mathcal{I}}$ (since $R_{p_1}(c_i, a_k) \in \mathcal{A}$ and $\mathcal{I}$ is a model of $\mathcal{A}$), and hence $V(a_k) = \mathit{true}$ and C_i has value true by using V. If the 1st literal of C_i is $\neg a_k$ for some $1 \leq k \leq l$, then $a_k^{\mathcal{I}} \in F^{\mathcal{I}}$ (since $R_{n_1}(c_i, a_k) \in \mathcal{A}$ and $\mathcal{I}$ is a model of $\mathcal{A}$), and hence $a_k^{\mathcal{I}} \notin T^{\mathcal{I}}$, and hence $V(a_k) = \mathit{false}$ and C_i has value true by using V. The cases when $c_i^{\mathcal{I}} \in L2^{\mathcal{I}}$ or $c_i^{\mathcal{I}} \in L3^{\mathcal{I}}$ can be handled in a similar way. Hence X is satisfied by V, which completes the proof. ◁

B Optimizations

We briefly discuss some optimizations for Algorithm 1 and the instance checking problem.

The loop at Step 2 of Algorithm 1 can be done only for $x \in \Delta$ that are reachable from Δ_0 (via a path using edges of $\mathcal{U}_0$). The reason is that, an iteration of that loop for

an x unreachable from Δ_0 does not affect the part of the model graph related with the elements of Δ that are reachable from Δ_0, and at the end only such a part is essential for the result of the algorithm.

Consider the procedure Simulate-Changing-Content(x, Γ), which is called in Algorithm 1. Observe that, for its calls in Algorithm 1, Γ depends only on x, $\mathbf{C}(x)$ and the selected formula $C \in \mathbf{C}(x)$ at Step 2 of Algorithm 1. The first instruction of that procedure set $x_* := \mathsf{Find}(\Gamma)$. If $x_* = x$ then, of course, the remaining instructions of the procedure do not have any effect and can therefore be ignored. In the case x_* is a newly created element (different from x), then the mentioned remaining instructions of the procedure are expected to be executed not only at the moment of the calling but also at later moments when there appear new connections via $\mathbf{E}$ or $\mathbf{U}$ to x. That is, we can remember x_* as a replacement of x, and whenever there appears a connection via $\mathbf{E}$ or $\mathbf{U}$ to x we can immediately direct the connection to x_* (instead of x).

Steps 3a and 3b of Algorithm 1 can be moved to Step 2 as a substep 2d for the case $C = \forall R_t.D$, assuming that the two occurrences of $\mathbf{C}(x)$ in those steps are changed to C. Assume that we have this modification. Then, to reduce redundant computation, when a formula C of $\mathbf{C}(x)$ has been processed at Step 2 we mark it as *resolved* if it is not of the form $D \sqsubseteq D'$. We will select a formula C of $\mathbf{C}(x)$ for consideration at Step 2 only if it is not resolved. Furthermore, when simulating the role of x by a newly created element x_* in an execution of Simulate-Changing-Content(x, Γ) at Step 2(a)ii or 2(b)ii:

- we can transfer the marks (*resolved* or the default value *unresolved*) of the formulas of $\mathbf{C}(x)$ to the formulas of $\mathbf{C}(x_*)$ (note that $\mathbf{C}(x) \subseteq \mathbf{C}(x_*)$);
- if $\mathbf{E}(x, \exists R_t.D) = z$ then we can set $\mathbf{E}(x_*, \exists R_t.D) := z$, for any $\exists R_t.D \in \mathbf{C}(x)$;
- if $\mathbf{U}(x, R_t) = z$ then we can set $\mathbf{U}(x_*, R_t) := z$, for any $t \in \mathrm{IND}$.

Similar modifications can be made for the pair y and y_* in the mentioned new step 2d (the case $C = \forall R_t.D$) if y_* is a newly created element of Δ.

The selection orders of $x \in \Delta$ (at Steps 2 and 3) and $C \in \mathbf{C}(x)$ (at Steps 2), as well as the execution order of Steps 2a, 2b, 2c, 2d (of the modified version) and 3c (of the original version) are not essential for correctness and the PTIME data complexity of the algorithm. However, they may affect efficiency of the algorithm. Different orders can be experimented with to find an efficient one. For example, one may give a preference to realizing requirements of the form $D \sqcap D'$, then creating connections of the form $\mathbf{U}(x, R_t) = y$, then realizing requirements of the form $\forall R_t.D$, then realizing requirements of the form $\exists R_t.D$ or $D \sqsubseteq D'$.

Given a Horn knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ in $\mathcal{R}eg$ and a positive formula C such that the constructors $\forall$ and $\forall\exists$ do not occur in C and the premises of the program clauses of $\mathcal{T}$, by using the relational translation (and Skolemization), one can translate the problem of checking $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$ to the problem of checking unsatisfiability of a set Γ of Horn clauses in predicate logic. Analyzing the dependencies of predicates, one may ignore certain clauses of Γ that are irrelevant for deriving the empty clause. This corresponds to a certain simplification of the knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ w.r.t. the query specified by $C(a)$. That is, analyzing dependency of predicates of Γ, one may reduce the knowledge base $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ to a smaller one $(\mathcal{R}', \mathcal{T}', \mathcal{A}')$ such that $(\mathcal{R}, \mathcal{T}, \mathcal{A}) \models C(a)$ iff $(\mathcal{R}', \mathcal{T}', \mathcal{A}') \models C(a)$.